PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2001-357071
(43)Date of publication of application : 26.12.2001

(51)Int.Cl. G06F 17/30
H04L 12/46

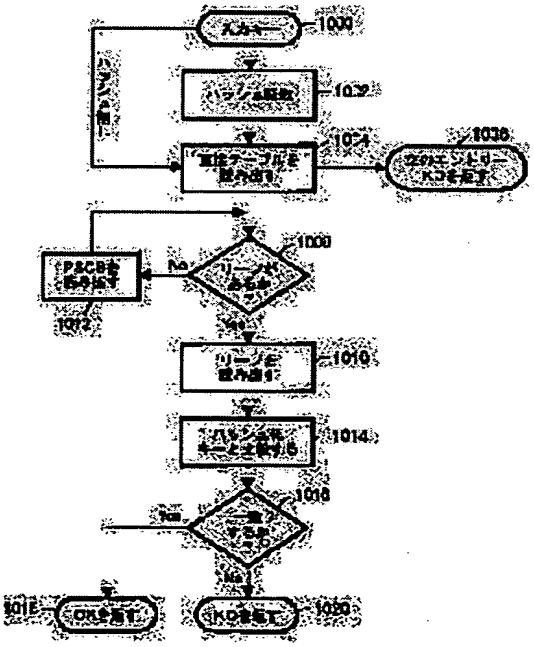
(21)Application number : 2001-091839
(22)Date of filing : 28.03.2001

(71)Applicant : INTERNATL BUSINESS MACH CORP <IBM>
(72)Inventor : BASS BRIAN MITCHELL
CALVIGNAC JEAN LOUIS
HEDDES MARCO C
ANTONIOS MARAGUKOSU
SIEGEL MICHAEL STEVEN
VERPLANKEN FABRICE JEAN

(30)Priority
Priority number : 2000 543531 Priority date : 06.04.2000 Priority country : US

(54) PACKAGING OF FULL MATCHING SEARCH ALGORITHM FOR NETWORK PROCESSOR

(57)Abstract:
PROBLEM TO BE SOLVED: To provide novel data structure, method and device for finding out full matching(FM) between a search pattern and a pattern stored in the leaf of a search tree.
SOLUTION: A key is inputted, a hash function is executed to a key, a direct table(DT) is accessed, and walk-through of the tree is performed until reaching the leaf through a pattern search control block(PSCB). Both the key and correspondent information required for retrieval are stored in a Patricia tree structure and the hash function performs mapping of n->n from the bit of the key to the bit of a hashed key.



LEGAL STATUS

[Date of request for examination] 28.03.2001
[Date of sending the examiner's decision of rejection]
[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]
[Date of final disposal for application]
[Patent number]
[Date of registration]
[Number of appeal against examiner's decision of rejection]

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号
特開2001－357071
（P2001－357071A）
(43)公開日　平成13年12月26日(2001.12.26)

(51)Int.Cl.[7]	識別記号	Ｆ Ｉ	テーマコード（参考）
Ｇ０６Ｆ　17/30	４１２	Ｇ０６Ｆ　17/30	４１２
	３５０		３５０Ａ
	４１４		４１４Ｚ
Ｈ０４Ｌ　12/46		Ｈ０４Ｌ　12/46	Ａ

審査請求　有　請求項の数34　ＯＬ　（全 25 頁）

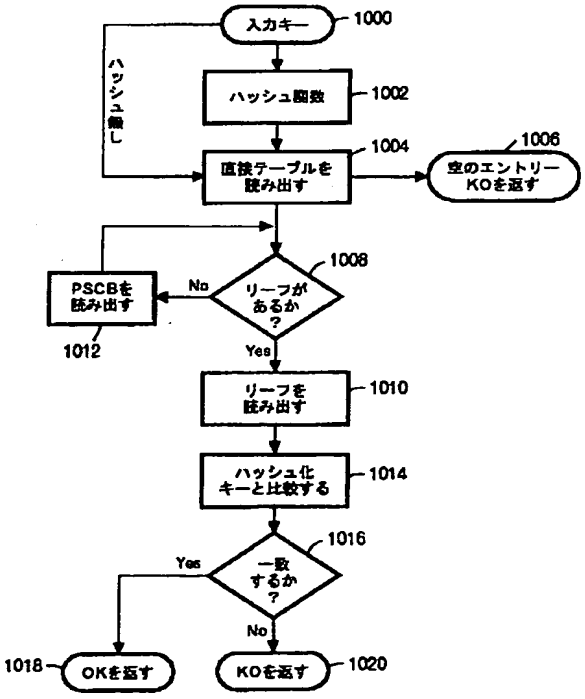
(21)出願番号　特願2001－91839(P2001－91839)
(22)出願日　平成13年３月28日(2001.3.28)
(31)優先権主張番号　０９／５４３５３１
(32)優先日　平成12年４月６日(2000.4.6)
(33)優先権主張国　米国（ＵＳ）

(71)出願人　390009531
インターナショナル・ビジネス・マシーンズ・コーポレーション
ＩＮＴＥＲＮＡＴＩＯＮＡＬ　ＢＵＳＩＮＥＳＳ　ＭＡＳＣＨＩＮＥＳ　ＣＯＲＰＯＲＡＴＩＯＮ
アメリカ合衆国10504、ニューヨーク州　アーモンク　（番地なし）
(74)代理人　100086243
弁理士　坂口　博　（外２名）

最終頁に続く

(54)【発明の名称】　ネットワーク・プロセッサ用の完全一致（ＦＭ）サーチ・アルゴリズムの実装

(57)【要約】　（修正有）
【課題】　サーチ・パターンとサーチ・ツリーのリーフに格納されたパターンとの間の完全一致を見つけるための新規のデータ構造、方法、および装置を提供する。
【解決手段】　キーを入力し、キーに対しハッシュ関数を実行し、直接テーブル（ＤＴ）にアクセスし、パターン・サーチ制御ブロック（ＰＳＣＢ）を通してリーフに達するまでツリーのウォーク・スルーを行う。キーおよび検索のために必要な対応する情報の両方が、パトリシア・ツリー構造に格納され、ハッシュ関数は、キーのビットからハッシュ・キーのビットへｎ－＞ｎのマッピングを行う。



1

【特許請求の範囲】

【請求項１】入力キーをサーチ文字列として読み出す動作と、

ハッシュ関数を使用して前記入力キーをハッシュ化してハッシュ化キーを生成する動作と、

前記ハッシュ化キーの最上位Ｎビットを、各々の非空エントリがサーチ・ツリーの次の分岐またはリーフのポインタを含むサーチ・ツリーの複数のルート・ノードを表すテーブルのインデックスとして使用する動作と、

非空テーブル・エントリのポインタが、対応するサーチ・ツリーのリーフまたは次の分岐を指し示すかどうかを決定する動作と、

ポインタが対応するサーチ・ツリーのリーフを指し示さない場合、次の分岐内容を読み出す動作と、

対応するサーチ・ツリーのリーフに達した場合、リーフ内容を読み出し、リーフのパターンを前記ハッシュ化キーと比較して、前記リーフ・パターンが前記ハッシュ化キーと一致するかどうかを決定する動作と、

前記リーフ・パターンが前記ハッシュ化キーと一致する場合、要求するアプリケーションに見つかったリーフの内容を返す動作とを含む、コンピュータ処理装置によって可変長サーチ・キーの完全一致を決定するための方法。

【請求項２】前記サーチ・ツリーの複数のルート・ノードを表すテーブルが2^N個のエントリを含む、請求項１に記載の完全一致を決定するための方法。

【請求項３】前記コンピュータ処理装置がネットワーク・プロセッサである、請求項１に記載の完全一致を決定するための方法。

【請求項４】前記対応するサーチ・ツリーの次の分岐の内容が別の次の分岐を指し示す、請求項１に記載の完全一致を決定するための方法。

【請求項５】前記次の分岐の内容が対応するサーチ・ツリーのリーフを指し示す、請求項１に記載の完全一致を決定するための方法。

【請求項６】前記リーフ・パターンが前記ハッシュ化キーと一致せず、かつ別のリーフのポインタを含まない場合、一致見つからずの標識を返すことをさらに含む、請求項１に記載の完全一致を決定するための方法。

【請求項７】テーブルのインデックスが空エントリのインデックスである場合、一致見つからずの標識を返すことをさらに含む、請求項１に記載の完全一致を決定するための方法。

【請求項８】色レジスタの内容を前記ハッシュ化キーに付加して最終ハッシュ化キーを提供することをさらに含む、請求項１に記載の完全一致を決定するための方法。

【請求項９】ゼロの文字列を前記ハッシュ化キーに付加して最終ハッシュ化キーを提供することをさらに含む、請求項１に記載の完全一致を決定するための方法。

【請求項１０】前記次の分岐のビット数が前記ハッシュ化キーの長さを超える場合、完全一致のサーチを終了する動作をさらに含む、請求項１に記載の完全一致を決定するための方法。

2

【請求項１１】前記入力キーに対して使用される前記ハッシュ関数が、前記ハッシュ化キーを前記入力キーに変換できる可逆ハッシュ関数である、請求項１に記載の完全一致を決定するための方法。

【請求項１２】前記リーフが別のリーフへのチェーン・ポインタを含む場合、別のリーフに格納されたパターンを読み出し、前記パターンを前記ハッシュ化キーと比較する動作と、

前記格納されたパターンが前記ハッシュ化キーと一致せず、かつ前記チェーンの次のリーフのポインタを含まない場合、一致見つからずの標識を返す動作とをさらに含む、請求項１に記載の完全一致を決定するための方法。

【請求項１３】前記リーフが別のリーフのチェーン・ポインタを含む場合、別のリーフに格納されたパターンを読み出し、前記パターンを前記ハッシュ化キーと比較する動作と、

前記格納されたパターンが前記ハッシュ化キーと一致する場合、一致発見の標識を返す動作とをさらに含む、請求項１に記載の完全一致を決定するための方法。

【請求項１４】サーチすべきパターンまたはキーと、

サーチ・ツリーの第１アドレス位置を格納する直接テーブルと、

各々がサーチ・ツリーの分岐を表す複数のパターン・サーチ制御ブロックと、

各リーフがサーチの結果のためのアドレス位置である複数のリーフとを含む、可変長サーチ・キーの完全一致を見つけるための複数のデータ構造を含むコンピュータ読み出し可能な媒体。

【請求項１５】ツリー・サーチ・メモリを管理するルックアップ定義テーブルをさらに含む、請求項１４に記載の完全一致を見つけるための複数のデータ構造を含むコンピュータ読み出し可能な媒体。

【請求項１６】前記ルックアップ定義テーブルは、ツリーが存在する物理的メモリ、キーおよびリーフのサイズ、および実行すべきサーチの種類を定義するエントリを含む、請求項１５に記載の完全一致を見つけるための複数のデータ構造を含むコンピュータ読み出し可能な媒体。

【請求項１７】前記ルックアップ定義テーブルが複数のメモリに実装される、請求項１４に記載の完全一致を見つけるための複数のデータ構造を含むコンピュータ読み出し可能な媒体。

【請求項１８】直接テーブル・エントリのフォーマットがサーチ制御ブロック、次のパターン・サーチ制御ブロックを指し示す次のパターン・アドレス、リーフまたは結果を指し示すリーフ制御ブロック・アドレス、次の試験ビット、および直接リーフのうちの少なくとも１つを

含む、請求項１４に記載の完全一致を見つけるための複数のデータ構造を含むコンピュータ読み出し可能な媒体。

【請求項１９】パターン・サーチ制御ブロックのフォーマットがサーチ制御ブロック、次のパターン・サーチ制御ブロックを指し示す次のパターン・アドレス、リーフまたは結果を指し示すリーフ制御ブロック・アドレス、および次の試験ビットのうちの少なくとも１つを含む、請求項１４に記載の完全一致を見つけるための複数のデータ構造を含むコンピュータ読み出し可能な媒体。

【請求項２０】リーフ・データ構造が１つのリーフ連鎖ポインタ、プレフィックス長、サーチ・キーと比較すべきパターン、および可変ユーザ・データのうちの少なくとも１つを含む、請求項１４に記載の完全一致を見つけるための複数のデータ構造を含むコンピュータ読み出し可能な媒体。

【請求項２１】前記直接リーフは直接テーブル・エントリに直接格納され、かつサーチ制御ブロックおよびサーチ・キーと比較されるパターンを含む、請求項１８に記載の完全一致を見つけるための複数のデータ構造を含むコンピュータ読み出し可能な媒体。

【請求項２２】パターン・サーチ制御ブロックが、サーチ・ツリーのリーフ・パターンが異なる位置に挿入される、請求項１４に記載の完全一致を見つけるための複数のデータ構造を含むコンピュータ読み出し可能な媒体。

【請求項２３】パターン・サーチ制御ブロックは１の幅および１の高さによって定義される形状を持ち、かつ少なくとも３６ビットのライン長を有するメモリに格納される、請求項１４に記載の完全一致を見つけるための複数のデータ構造を含むコンピュータ読み出し可能な媒体。

【請求項２４】フレーム処理を行う複数のプロトコル・プロセッサおよび内部制御点プロセッサを含む埋込み式プロセッサ複合体と、

各プロトコル・プロセッサにアクセスでき、高速パターン・サーチ、データ操作、およびフレーム・パージングをもたらす複数のハードウェア・アクセラレータ・コプロセッサと、

少なくとも１つのサーチ・ツリーを表す複数のデータ構造であって、直接テーブルとパターン・サーチ制御ブロックとリーフとを含む前記データ構造を格納する複数のプログラム可能なメモリ装置と、

各プロトコル・プロセッサの複数のメモリ装置へのアクセスを制御する制御メモリ・アービタとを含む、可変長サーチ・キーの完全一致を決定するために半導体基板上に組み立てられた装置。

【請求項２５】プロトコル・プロセッサの実行と並行して作動して、メモリの読み書きおよびメモリ範囲検査を含むツリー・サーチ命令を実行するツリー・サーチ・エンジンをさらに含む、請求項２４に記載の完全一致を決定するために半導体基板上に組み立てられた装置。

【請求項２６】前記複数のメモリ装置は内部スタティック・ランダム・アクセス・メモリ、外部スタティック・ランダム・アクセス・メモリ、および外部ダイナミック・ランダム・アクセス・メモリの少なくとも１つをさらに含む、請求項２４に記載の完全一致を決定するために半導体基板上に組み立てられた装置。

【請求項２７】前記制御メモリ・アービタが、複数のプロトコル・プロセッサと複数のメモリ装置との間でメモリ・サイクルを割り当てることによって、制御メモリ動作を管理する、請求項２４に記載の完全一致を決定するために半導体基板上に組み立てられた装置。

【請求項２８】各プロトコル・プロセッサは一次データ・バッファ、スクラッチ・パッド・データ・バッファ、およびデータ格納動作のための制御レジスタを含む、請求項２４に記載の完全一致を決定するために半導体基板上に組み立てられた装置。

【請求項２９】サーチ・キーに対して幾何学的ハッシュ関数を実行するハッシュ・ボックス・コンポーネントをさらに含む、請求項２４に記載の完全一致を決定するために半導体基板上に組み立てられた装置。

【請求項３０】プログラム可能なサーチ・キー・レジスタおよびプログラム可能なハッシュ化キー・レジスタをさらに含む、請求項２４に記載の完全一致を決定するために半導体基板上に組み立てられた装置。

【請求項３１】複数の独立サーチ・ツリーの間で単一のデーブル・データ構造を共用することを可能にするプログラム可能な色キー・レジスタをさらに含む、請求項３０に記載の完全一致を決定するために半導体基板上に組み立てられた装置。

【請求項３２】色レジスタがイネーブルされている場合、その内容をハッシュ出力に付加して最終ハッシュ化キーを生成する、請求項３１に記載の完全一致を決定するために半導体基板上に組み立てられた装置。

【請求項３３】色レジスタがイネーブルされていない場合、同等の数のゼロをハッシュ出力に付加して最終ハッシュ化キーを生成する、請求項３１に記載の完全一致を決定するために半導体基板上に組み立てられた装置。

【請求項３４】入力キーをサーチ文字列として読み出すプログラム命令と、

ハッシュ化キーを入力キーに変換することができる可逆ハッシュ関数を使用して前記入力キーをハッシュ化するプログラム命令と、

前記ハッシュ化キーの最上位Ｎビットを、各非空エントリがサーチ・ツリーの次の分岐またはリーフへのポインタを含むサーチ・ツリーの複数のルート・ノードを表すテーブルのインデックスとして使用するプログラム命令と、

非空テーブル・エントリのポインタが対応するサーチ・ツリーのリーフまたは次の分岐を指し示すかどうかを決

定するプログラム命令と、
ポインタが対応するサーチ・ツリーのリーフを指し示さない場合、次の分岐の内容を読み出すプログラム命令と、
対応するサーチ・ツリーのリーフに達したときにリーフ内容を読み出し、リーフのパターンをハッシュ化キーと比較して、リーフ・パターンがハッシュ化キーと一致するか否かを決定するプログラム命令と、
リーフ・パターンがハッシュ化キーと一致する場合、要求するアプリケーションに見つかったリーフの内容を返すプログラム命令とを含む、可変長サーチ・キーの完全一致を決定するためのコンピュータ・プログラム。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、一般的に、パターン・マッチング・アルゴリズムに関し、さらに詳しくは、ネットワーク処理装置に実装することができる完全マッチング・サーチ・アルゴリズムに関する。

【０００２】

【従来の技術】ますます複雑になるタスクをメディア速度でサポートするためのハードウェア統合処理の要求から、ネットワーク・プロセッサが生成された。ネットワーク・プロセッサは、１組の埋込み型プログラム可能プロトコル・プロセッサおよび相補型システム・コプロセッサにより機能柔軟性を備えた配線速度のフレーム処理および転送能力を提供する。ネットワーク・プロセッサは、今日のパーソナル・コンピュータにとってのマイクロプロセッサと同様に、ネットワークにとっての基本的ネットワーク構築ブロックになるものと期待される。ネットワーク・プロセッサは複数のデータ・ストリームの実時間処理を提供し、セキュリティの強化やＩＰパケット処理および転送能力をもたらす。さらに、それは、並列分散処理およびパイプライン処理設計などの高度アーキテクチャにより、速度改善をもたらす。これらの能力は効率的なサーチ・エンジンを可能にし、データ処理スループットを向上し、複雑なタスクの高速実行をもたらす。ネットワーク・プロセッサのプログラム可能な機能は、ネットワーク製品開発者に、新しい特注のアプリケーション特定的な集積回路（ＡＳＩＣ）の設計を必要とすることなく、新しいプロトコルおよび技術を実装するためのより容易な移行経路を提供する。

【０００３】ネットワーク・プロセッサは、インターネット・プロバイダまたはエンタープライズ・ネットワーク・プロバイダ向けの相互接続解決策を開発するための高度にカスタマイズ可能でスケーラブルな技術を提供する。ネットワーク・プロセッサは、ローエンド独立型装置から大型マルチラック解決策まで広範囲の解決策の基礎を提供する。この性質の規模変更は、高性能非ブロッキング・パケット・ルーティング・スイッチ技術、および他の産業スイッチ技術に適応できるＩＢＭ社のData A

ligned Serial Link（ＤＡＳＬ）インタフェースなど専有技術のインタフェースを使用することにより達成される。

【０００４】プログラム可能な通信集積回路として、ネットワーク・プロセッサは、非常に効率的なパケット分類、フレーム毎のマルチテーブル・ルックアップ、パケット変更、キュー／ポリシー管理、およびその他のパケット処理能力を提供する。ネットワーク・プロセッサは、スイッチング・エンジン、サーチ・エンジン、フレーム・プロセッサ、およびイーサネット（登録商標）ＭＡＣを１つの装置に統合して、いずれかのプロトコル層でフレーム内容に基づき高容量メディア重みスイッチング・フレームを必要とする顧客の要求をサポートする。

【０００５】ハードウェア・アクセラレータは、フレーム転送、フレーム・フィルタリング、およびフレーム変更を実行する。複合範囲および動作仕様を持つ何百ものルールを実施するネットワーク・プロセッサの能力は、フィルタリング能力の新しいベンチマークを設定し、ネットワーク・プロセッサに基づくシステムを高容量サーバ・ファーム・アプリケーションに独特に適合させる。

【０００６】ネットワーク・プロセッサを用いて展開される典型的なシステムは、分散型ソフトウェア・モデルを使用し、各プログラム可能なネットワーク・プロセッサがタスクを同時に実行する。幾つかの機能は制御点（ＣＰ）プロセッサで実行され、これはネットワーク・プロセッサの内部または外部に置くことができる。ＣＰは、レイヤ２およびレイヤ３のルーティング・プロトコル、ならびにレイヤ４およびレイヤ５のネットワーク・アプリケーションおよびシステム管理をサポートする。配線速度の転送およびフィルタリング機能は、ネットワーク・プロセッサ・ハードウェアおよび常駐ピココードの組合せによって実行される。

【０００７】多数の相互接続されたノードを含む通信ネットワークで、データは１つのノードから他のいずれかのノードまたはネットワークに送信することができる。ルータと呼ばれる特殊化されたノードは、データをその宛先に転送する能力を持つ。通信ネットワークを通して送信されるデータは、一般的にヘッダの一部として、宛先アドレスに関する情報を含む。各ルータはこの情報または少なくともその一部を、内部に格納されたアドレスのリストと比較する。格納されたアドレスと宛先アドレスとの間に一致が見つかると、ルータはその宛先ノードに通じる経路を確立する。ネットワークの規模および構造によって、データはその宛先に直接転送されるか、または他の中間ルータへ送信される。国際標準化機構（ＩＳＯ）は、ルータが部分アドレスのルーティング情報を格納するルーティング規格を発表した。ルータはそこで、そのデータベースにある最もよく一致する部分アドレスにパケットを送信する。ＩＳＯ規格は、所定の桁数または所定のヘッダ長を使用してノードの階層構造を構

築することを可能にする。主ルータはアドレスの最初の部分によってアドレス指定され、サブルータは中間部分によって、かつ最終宛先はアドレスの最後の数桁によって指定される。したがって、いずれかのルータが、データの送信先の階層のレベルに割り当てられた桁を読み取れば充分である。

【０００８】受信パケットのルーティングは、付随するアドレス文字列に基づく。アドレス文字列は、アドレス文字列と共に、どのルータがパケットの配信における次のルータであるかなど他の関連詳細を含むデータベースで、サーチ・キーとして使用される。データベースはルーティング・テーブルと呼ばれ、現在のルータと次のルータとの間のリンクは、パケットの進行における次のホップと呼ばれる。ルーティング・テーブル・サーチ・プロセスは、アドレスの構造のみならず、テーブルの編成にも依存する。たとえば非階層構造を有する７ビット以下のサイズのサーチ・キーは、一連のアドレス・エントリとして編成されたルーティング・テーブルでは最も効率的に見つかる。サーチ・キーは、テーブルで正しいエントリをつき止めるための索引として使用される。より大きいサイズ、たとえば３２ビットのサーチ・キーの場合、対応するルーティング・テーブルは、１０，０００個を超えるエントリを持つことができる。データベースを索引によって直接サーチする単純なテーブルとして編成すると、テーブルの大部分が空になるので、大量のメモリ空間を無駄にすることになる。

【０００９】従来のルータは、サーチ・プロセスを幾つかのステップに分割する。第１ステップは、ルータが宛先ホスト・コンピュータに直接接続されているかどうかを決定することである。この場合、メッセージは宛先から１ホップであり、その方向に経路選択しなければならない。宛先コンピュータがそのルータに直接接続されていない場合、次のステップは、宛先ネットワークのトポロジカル方向を決定することである。トポロジカル・レイアウトから方向が決定されると、メッセージはその方向に経路選択される。そうでない場合、最終ステップは、メッセージをデフォルト・リンクに沿って経路選択することである。

【００１０】一般的に、第１ステップは、ルータに直接接続された３２ビットのアドレスのホスト・コンピュータを含むテーブル内の線形サーチを使用して実行される。ローカル・トポロジを反映して、アドレス・テーブル内の各エントリは、アドレス指定されたコンピュータに直接通じている対応する出力インタフェースに接続される。宛先アドレスをルータが受け取ると、３２ビット全部が、テーブル内の宛先アドレスの各々と比較される。一致が見つかると、メッセージは、指定されたルータ・インタフェースを介して、対応する宛先に直接送信される。

【００１１】第２ステップ、つまり宛先ネットワークの

方向を決定するステップは通常、ネットワーク・アドレスの数のためにそのようなテーブルの管理および使用が困難になるので、テーブル内の線形サーチによって行われることはない。従来の技術では、アドレス文字列がネットワーク・アドレス、サブネット・アドレス、およびホスト識別の３つの階層レベルに一致すると、ルータは、ハッシュ化、パトリシア・ツリー・サーチ、およびマルチレベル・サーチなど、幾つかの周知の技術の１つを使用して決定を行った。ハッシュ化で、ハッシュ関数はアドレスのネットワーク部分を縮小し、小さい管理可能な索引を生成する。ハッシュ索引は、ハッシュ・テーブルに索引を付け、一致するハッシュ・エントリを探索するために使用される。ハッシュ・テーブルの各ハッシュ・エントリに対応するのが、対応するネットワークのトポロジカル方向を指し示す出力インタフェースのアドレスである。ハッシュ・ネットワーク部分とハッシュ・エントリとの間に一致が見つかると、メッセージは、対応するインタフェースおよび宛先ネットワークの方向に向けられる。

【００１２】ハッシュ化により、大きい管理不能なフィールドは、小さい管理可能な索引に縮小される。しかし、このプロセスでは、２つまたはそれ以上のフィールドから同一ハッシュ索引が生成されることがある。これらのフィールドはハッシュ・テーブルの同一場所に格納しなければならないので、このできごとは衝突と呼ばれる。衝突中にエントリを区別するために、さらなるサーチが必要である。したがって、衝突は、ハッシュ・サーチを用いることから得られる効率を低下し、全ての許容アドレスが単一の索引に縮小される最悪の場合には、ハッシュ化はサーチ・プロセスとして事実上役立たないとみなされる。

【００１３】パトリシア・ツリー・サーチは、ハッシュ法によって生じる衝突を回避する。このサーチ法は、全てのアドレス文字列およびたとえば関連ルート情報などの付随情報をバイナリ・ツリーに格納する必要がある。このサーチ・プロセスは、アドレスをアドレス文字列内の最上位ビット位置から１ビットずつ、ツリー・ノードと比較する。一致したビット値は、左または右子ノードのいずれかに進むようにサーチを誘導し、このプロセスがアドレスの次のビットに対して繰り返される。サーチ時間は、格納された最長アドレス文字列のサイズに比例する。パトリシア・ツリー・サーチでは、平均サーチ時間と最悪の場合のサーチ時間との間の差は、あまり大きくない。さらに、ルーティング・テーブルは極めて効率的に編成される。それは、ハッシュ法の匹敵するルーティング・テーブルより少ないメモリを必要とする。パトリシア・ツリー・サーチは、最悪の場合のサーチをハッシュ法よりよく処理するが、大抵の場合、一致をつき止めるまでの時間がかなり長くかかる。したがって、多くの従来のルータは、ハッシュ化とパトリシア・ツリー・

9

サーチの組合せを使用する。この組合せはマルチレベル・サーチと呼ばれる。

【００１４】マルチレベル・サーチは、ハッシュ化とパトリシア・ツリー・サーチを結合したものである。キャッシュは、最も最近に、かつおそらく最も一般的に経路選択されたネットワーク・アドレスの部分セットを含むハッシュ・テーブルを格納し、パトリシア・ツリーはネットワーク・アドレスの全セットを格納する。メッセージを受信すると、宛先アドレスはテーブル上にハッシュ化される。それが予め定められた時間内につき止められない場合、アドレスは、アドレスが格納されていれば見つかることを確実にするパトリシア・ツリー・サーチ・エンジンに渡される。

【００１５】従来の技術には、固定マッチ・ツリー、最長プレフィックス・マッチ・ツリー、およびソフトウェア管理ツリーをはじめ、多数の既知のツリー・サーチ・アルゴリズムがある。固定マッチ・ツリーは、レイヤ２イーサネットＭＡＣテーブルなど、厳密な一致を要求する固定サイズ・パターンに使用される。最長プレフィックス・マッチ・ツリーは、ＩＰサブネット転送など、部分一致だけを必要とする可変長パターンに使用される。ソフトウェア管理ツリーは、フィルタ・ルールなど範囲またはビット・マスクとして定義されるパターンに使用される。一般的に、ルックアップはツリー・サーチ・エンジン（ＴＳＥ）を用いて実行される。

【００１６】

【発明が解決しようとする課題】本発明の目的は、パトリシア・ツリーのための完全一致ツリー・サーチ・アルゴリズムをハードウェア内に実装することである。このアルゴリズムの目的を果たすことができるようにメモリ構造をどのようにセットアップし、これらの構造をハードウェアがどのように処理するかを記載する。

【００１７】本発明の別の目的は、以前のポインタに関する記憶を必要とせず、試験する次のビットまたはビット群と共に正方向ポインタのみを使用し、それによりノードの記憶空間を縮小するサーチ機構を提供することである。

【００１８】

【課題を解決するための手段】主要な概念は、キーを入力し、キーに対してハッシュ関数を実行し、直接テーブル（ＤＴ）にアクセスし、パターン・サーチ制御ブロック（ＰＳＣＢ）を通してツリーのウォーク・スルーを実行して、リーフで終了するというものである。

【００１９】解決される問題は、少数のレジスタおよび正規メモリに配置することができ、次いで比較的単純なハードウェア・マクロによって操作できるパトリシア・ツリー構造を構築するために使用できる、１組のデータ構造の設計である。パトリシア・ツリーに、キーおよび検索のために必要な対応する情報の両方が格納される。

【００２０】キーは、探索され照合される情報である。

10

最初に、キーがレジスタに入れられ、ハッシュ化される。その結果がハッシュ・キーであり、実際のサーチはハッシュ・キーに対して行われる。ハッシュ関数は空ハッシュとすることができ、その場合ハッシュ・キーはキーと全く同一となる。ハッシュ関数は、キーのビットからハッシュ・キーのビットへｎ－＞ｎのマッピングを行う。

【００２１】ハッシュ・キーおよび関連情報をツリーに格納するために使用されるデータ構造は、リーフと呼ばれる。リーフを検索することが、このアルゴリズムの目的である。各リーフは、入力キーと厳密に一致する単一のキーに対応する。この実現ではリーフはキーを含み、格納すべき追加情報がそれに付加される。リーフの長さは、キーの長さの場合と同様に、プログラム可能である。リーフはランダム・アクセス・メモリに格納され、単一メモリ・エントリとして実現される。キーが直接テーブル（ＤＴ）内にある場合には、それは直接リーフと呼ばれる。

【００２２】

【発明の実施の形態】本発明について、本発明が埋め込まれたネットワーク・プロセッサの文脈で説明する。ネットワーク・プロセッサ１０は、単一チップ上のプログラム可能なスイッチングおよびルーティング・システムであり、そのアーキテクチャを図１に示す。それは、スイッチ・インタフェースに接続するためのData Aligned Serial Link（ＤＡＳＬ）のみならず、１０／１００イーサネット、ギガビット・イーサネット、およびPacket Over SONET（ＰＯＳ）用のメディア・インタフェースを提供する。内部ハードウェア・アクセラレータは、性能および効率を高める。埋込み型プロセッサ複合体（ＥＰＣ）１２は、プロトコル・プロセッサおよびフレーム処理、構成、および管理サポートのための内部制御点プロセッサを含む。

【００２３】最高Ｎ個までの並列プロトコル・プロセッサが利用可能である。１６個のプロトコル・プロセッサの実施形態の場合、１６，３８４語の内部ピココード命令記憶装置および３２，７６８語の外部ピココード命令記憶装置が利用可能である。毎秒２１２，８００万個の命令（ＭＩＰＳ）の集合処理能力を提供することができる。さらに、各プロトコル・プロセッサは、高速パターン・サーチ、データ操作、内部チップ管理機能、フレーム・パージング、およびデータ先取りサポートを提供する、Ｍ個のハードウェア・アクセラレータ・コプロセッサにアクセスできる。好ましい実施形態では、プロトコル・プロセッサ用の制御記憶装置は、内部および外部メモリの両方、すなわち即時アクセス用の３２Ｋの内部スタティック・ランダム・アクセス・メモリ（ＳＲＡＭ）２８、高速アクセス用の外部ゼロ・バス・ターンアラウンド（ＺＢＴ）ＳＲＡＭ３０、および大容量記憶要求用の外部二重データ・レート（ＤＤＲ）ダイナミック・ラ

11

ンダム・アクセス・メモリ（ＤＲＡＭ）３２によって提供される。

【００２４】接続された制御点プロセッサ３４で作動する前処理アルゴリズムと共に埋込み型ハードウェア・アクセラレータを使用して、ネットワーク・プロセッサ１０は、複合範囲、優先順位、および動作仕様を持つ１００またはそれ以上のフィルタ・ルールにより、配線速度でフレームを処理することができる。これにより、ネットワーク・プロセッサに基づくシステムは、ゲートウェイ、サーバ・ファーム・アプリケーション、およびトラヒックの混合の処理に関連するフィルタリング・タスクによく適合するようになる。

【００２５】ネットワーク管理者がコヒーレントでユーザ・フレンドリーなインタフェースにフィルタ・ルールを入力すると、制御点ソフトウェアは自動論理検査を行う。安定度論理に基づき新規のフロー制御を使用して、ネットワーク・プロセッサ１０は、伝送制御プロトコル（ＴＣＰ）のコラプス無く、一般的に使用されるランダム早期廃棄法より高速の一時オーバサブスクリプションに耐える。ネットワーク・プロセッサ１０はまた、帯域幅を自動的に割り当てることによって、Differentiated Servicesをも配信し、ネットワーク管理者は、瞬時または仮定トラヒック統計に基づき多数の閾値を設定する効果を予測する必要性から解放される。

【００２６】単一ネットワーク・プロセッサ１０は、最高４０までの高速イーサネット・ポートまたは４つまでのギガビット・イーサネット・ポートのメディア速度スイッチングを行う。それはまた、ＯＣ－４８ｃ、ＯＣ－４８、４つのＯＣ－１２、または１６のＯＣ－３ポートをサポートするように構成することもできる。スケーラビリティのために、２つの３．５ＧｂｐｓシリアルＤＡＳＬリンクを使用して、２つのネットワーク・プロセッサを相互接続してポート密度を倍加するか、またはスイッチ構造を接続して最高６４までのネットワーク・プロセッサによるスイッチング解を生成することができる。システムのアベイラビリティを高めるため、２つのＤＡＳＬリンク、つまり１つの一次および１つの二次ＤＡＳＬリンクを、冗長スイッチ構造に接続することもできる。

【００２７】ネットワーク・プロセッサ１０の１つの例示的実施形態は、図１に示す通り次の主要部分を含む。

１．最高１６までのプログラム可能なプロセッサとコプロセッサを含む埋込み型プロセッサ複合体（ＥＰＣ）１２

２．フレームがイーサネット物理層装置からスイッチ構造へ移動するためのエンキュー・デキュー・スケジューリング論理１４（ＥＤＳイングレス）

３．フレームがスイッチ構造からイーサネット物理層装置へ移動するためのエンキュー・デキュー・スケジューリング論理１６（ＥＤＳエグレス）

12

４．別のネットワーク・プロセッサまたは中間スイッチへの相互接続のためのイングレス・スイッチ・インタフェース（スイッチ・イングレス）１８およびエグレス・スイッチ・インタフェース（スイッチ・エグレス）２０のＤＡＳＬリンク

５．イーサネットまたはＰＯＳ物理層装置２６からフレームを受信する物理ＭＡＣマルチプレクサ２２（ＰＭＭイングレス）、およびイーサネットまたはＰＯＳ物理層装置２６へフレームを送信する物理ＭＡＣマルチプレクサ２４（ＰＭＭエグレス）

【００２８】図２は、埋込み型プロセッサ複合体の例示的実施形態を示す。それは１６のプロトコル・プロセッサを含み、２１２８ＭＩＰＳの処理力を提供する。各プロトコル・プロセッサ４０は、３ステージ・パイプライン（取出し、復号、および実行）、汎用レジスタ、専用レジスタ、８命令キャッシュ、専用論理演算装置（ＡＬＵ）、およびコプロセッサを含み、全て１３３ＭＨｚで作動する。プロトコル・プロセッサのうち２つは特殊化されている。つまり、１つは誘導フレームの処理用であり（誘導フレーム・ハンドラ）、１つは制御メモリ内のルックアップ・データの構築用である（一般ツリー・ハンドラ）。

【００２９】図３は、プロトコル・プロセッサの例示的実施形態を示す。各々のプログラム可能なプロトコル・プロセッサ４０に関連付けられるコプロセッサは、次の機能を果たす。

１．データ記憶コプロセッサ６４は、フレーム・バッファ・メモリ４２、４４（イングレスおよびエグレス方向）をインタフェースして直接メモリ・アクセス（ＤＭＡ）機能をもたらす。

２．チェックサム・コプロセッサ６２は、ヘッダのチェックサムを計算する。

３．エンキュー・コプロセッサ６６は、キー・フレーム・パラメータを含む２５６ビット作業レジスタへのアクセスを制御する。このコプロセッサは完了ユニット４６とインタフェースして、フレームをスイッチおよびターゲット・ポート・キューに入れる。

４．インタフェース・コプロセッサは、デバッグまたは統計情報の収集のため、全てのプロトコル・プロセッサを内部レジスタ、カウンタ、およびメモリにアクセスさせる。

５．文字列コピー・コプロセッサは、ＥＰＣ内におけるデータの効率的な移動を可能にする。

６．カウンタ・コプロセッサは、プロトコル・プロセッサ４０のためのカウンタの更新を管理する。

７．ポリシー・コプロセッサはフロー制御情報を検討し、事前に割り当てられた帯域幅と一致するか検査する。

【００３０】ハードウェア・アクセラレータ４８は、フレーム転送、フレーム・フィルタリング、フレーム変

13

更、およびツリー・サーチを実行する。ネットワーク・プロセッサに組み込まれるその他の機能として、革新的フィルタ・ルール処理、ハッシュ関数、およびフロー制御が含まれる。

【００３１】プロトコル・プロセッサ４０は、複合範囲および動作仕様を持つ１００またはそれ以上のフレーム・フィルタ・ルールを実施することができる。フィルタリングはネットワーク・セキュリティのために不可欠であり、ネットワーク・プロセッサ・ハードウェア補助機構４８は、これらの複合ルール・セットを配線速度で実施する。フィルタ・ルールは、ＩＰヘッダ情報に基づき、フレームを拒絶または許可し、あるいはサービスの品質（ＱｏＳ）を割り当てることができる。ルールを前処理するための制御点ソフトウェアは、論理エラーを自動的に訂正する。論理的に正しいルール・セットが入力された後、パケット・ヘッダ情報からキーが形成され、ネットワーク・プロセッサのソフトウェア管理されるツリーを用いて、配線速度で試験される。

【００３２】幾何学的ハッシュ関数はＩＰヘッダの統計的構造を利用して、理想的ランダム・ハッシュより優れた性能を発揮する。したがって、低衝突率は、追加的解決サーチ無しで、完全一致テーブルの高速ルックアップを可能にする。

【００３３】プロトコル・プロセッサの実行と並行して作動して、ツリー・サーチ・エンジン７０はツリー・サーチ命令（メモリ読出し、書込み、または読み書き）、メモリ範囲検査、および不正メモリ・アクセス通知を実行する。図１４は、ツリー・サーチ・エンジンの例示的実施形態を示す。

【００３４】ネットワーク・プロセッサ１０内で２つのシステム制御オプションが利用可能である。内部プロセッサ３４は、システムの制御点（ＣＰ）プロセッサとして機能することができ、または代替的に、初期化および構成のために４つのイーサネット・マクロの１つに外部プロセッサ接続することができる。ＣＰプロセッサ３４は、誘導フレームと呼ばれる特殊なイーサネット・フレームを構築することによって、ネットワーク・プロセッサ内の他のプロセッサ・エンティティと通信する。誘導フレームはＤＡＳＬリンクをまたいで他の装置へ転送することができ、単一のイーサネット・ポートに接続された１つのＣＰプロセッサが、サブシステム内に含まれる全てのネットワーク・プロセッサ装置と通信してそれらを制御することを可能にする。各ネットワーク・プロセッサ１０の内部プロセッサ３４はまた、別個の３２ビットＰＣＩバスを用いて通信することもできる。

【００３５】ネットワーク・プロセッサ１０は通常サブシステム・ボード上にあり、プロトコル層（つまりレイヤ２、レイヤ３、レイヤ４およびより高いレイヤ）のフレーム処理を行う。ＣＰサブシステム内のＣＰプロセッサ３４上で作動するソフトウェアは、管理および経路発

14

見機能を提供する。ＣＰコード、プロトコル・プロセッサ上で作動するピココード、および誘導フレーム・ハンドラ上で作動するピココードは、このシステムの初期化、転送経路の維持、およびシステムの管理を可能にする。分散システムとして、ＣＰおよび各ネットワーク・プロセッサ・サブシステムは、効率および性能の向上のため、誘導フレームを使用して並行して作動したり通信する複数のプロセッサを含む。

【００３６】データ・フレームはＰＭＭ２２によってメディアから受信され、データ記憶バッファ４２に転送される。ＰＭＭはまた、受信プロセス中にＣＲＣ検査およびフレーム妥当性検証をも実行する。ディスパッチャ５０は、フレーム・ルックアップのために利用可能なプロトコル・プロセッサ４０に最高６４バイトまでのフレーム情報を送信する。分類ハードウェア補助機構４８は制御データを供給して、フレーム・フォーマットを識別する。プロトコル・プロセッサ４０は制御データを使用して、固定マッチ・ツリー、最長プレフィックス・マッチ・ツリー、またはソフトウェア管理ツリーをはじめ、適用すべきツリー・サーチ・アルゴリズムを決定する。

【００３７】ルックアップは、ツリー・サーチ・エンジン（ＴＳＥ）７０を用いて実行される。ＴＳＥ７０は制御メモリ７２へのアクセスを実行し、プロトコル・プロセッサ４０が実行を続けることを可能にする。制御メモリ７２は全てのテーブル、カウンタ、およびピココードによって必要とされるその他のデータを記憶する。効率のために、制御メモリ・アービタ５２は、プロトコル・プロセッサ４０と様々なオンチップおよびオフチップ制御メモリ・オプション５４との間でメモリ・サイクルを割り当てることによって、制御メモリ動作を管理する。

【００３８】プロトコル・プロセッサ４０は、データ記憶動作のために、一次データ・バッファ、スクラッチ・パッド・データ・バッファ、および制御レジスタ（集合的に７２）を含む。ひとたび一致が見つかると、ＶＬＡＮヘッダ挿入またはオーバレイなど、イングレス・フレーム変更を適用することができる。これらの変更は、ＥＰＣ１２によっては実行されない。代りに、ハードウェア・フラッグがセットされた場合に、イングレス・スイッチ・インタフェース・ハードウェア１８がこの変更を実行する。他のフレーム変更は、ピココードおよびデータ記憶コプロセッサ６４により、イングレス・データ記憶装置４２内に保持されたフレーム内容を変更することによって、達成することができる。

【００３９】フレームをスイッチ構造に送る前に、スイッチ・ヘッダおよびフレーム・ヘッダを構築するために、制御データが収集され使用される。制御データは、フレームの宛先などのスイッチ情報のみならず、エグレス・ネットワーク・プロセッサのための情報をも含み、それが宛先ポートのフレーム・ルックアップ、マルチキャストまたはユニキャスト・オペレーション、およびエ

15

グレス・フレーム変更を促進するのを助ける。

【００４０】図４は、例示的なイングレスおよびエグレス・フレームの流れを示す。完了後、エンキュー・コプロセッサ６６は、フレームをキュー制御ブロック（ＱＣＢ）７４に入れるために必要なフォーマットを作成し、それらを完了ユニット４６に転送する。完了ユニット４６は、最高１６のプロトコル・プロセッサ４０からスイッチ構造キュー７６へのフレームの順序を保証する。スイッチ構造キュー７６からのフレームは、それらがスイッチ構造７６によって転送されるときに、スイッチ・ヘッダおよびフレーム・ヘッダ・バイトを挿入して、６４バイト・セルに区分化される。

【００４１】スイッチ構造７６から受信したフレームは、リアセンブリ制御ブロック（ＲＣＢ）８０およびＥＤＳエグレス４４によって提供される情報を使用して、エグレス・データ記憶バッファ７８に入れられ、ＥＰＣ１２に登録される。フレームの一部分は、フレーム・ルックアップを実行するために、ディスパッチャ５０によってアイドル状態のプロトコル・プロセッサ４０に送られる。フレーム・データは、分類ハードウェア補助機構４８からのデータと共に、プロトコル・プロセッサ４０にディスパッチされる。分類ハードウェア補助機構４８は、イングレス・ネットワーク・プロセッサによって形成されたフレーム制御データを使用して、エグレス処理のための開始命令アドレスを決定するのに役立つ。

【００４２】エグレス・ツリー・サーチは、イングレス・サーチのためにサポートされるのと同じアルゴリズムをサポートする。ルックアップはＴＳＥ７０で行われ、プロトコル・プロセッサ４０は自由に実行を続けることができる。全ての制御メモリの動作は、プロセッサ複合体の間でメモリ・アクセスを割り当てる制御メモリ・アービタ５２によって管理される。

【００４３】エグレス・フレーム・データは、データ記憶コプロセッサ６４を通してアクセスされる。成功したルックアップの結果は、転送情報、および場合によってはフレーム変更情報を含む。エグレス・フレーム変更は、ＶＬＡＮヘッダの削除、活動時間の増分（ＩＰＸ）または減分（ＩＰ）、ＩＰヘッダ・チェックサムの再計算、イーサネット・フレームＣＲＣオーバレイ、およびＭＡＣ宛先アドレスまたはソース・アドレスのオーバレイまたは挿入を含むことができる。ＩＰヘッダ・チャックサムは、チェックサム・コプロセッサ６２によって作成される。変更が埋込み型プロセッサ複合体１２によって実行されるのではなく、ハードウェア・フラッグが形成され、ＰＭＭエグレス・ハードウェア２４が変更を実行する。完了後、フレームをＥＤＳエグレス・キュー４４に登録するために、エンキュー・コプロセッサ４６を使用して必要なフォーマットが構築され、それらが完了ユニット４６に転送される。完了ユニット４６は、最高１６のプロトコル・プロセッサから、エグレス・イーサネットＭＡＣに供給するＥＤＳエグレス・キュー４４へのフレーム順序を保証する。完成したフレームは最終的に、ＰＭＭエグレス・ハードウェア２４によってイーサネットＭＡＣまたはＰＯＳインタフェースに送られ、物理ポートから送り出される。

16

【００４４】図１４に示すように、ツリー・サーチ・エンジン（ＴＳＥ）７０は、ツリーの概念を使用して情報を格納したり検索する。検索つまりツリー・サーチのみならず、挿入や削除も、たとえばＭＡＣソース・アドレス、またはＩＰソース・アドレスとＩＰ宛先アドレスの連結などのビット・パターンであるキーに基づいて行われる。本発明で使用するための例示的ツリー・データ構造１００を図５に示す。情報は、少なくともキー１０２を含むリーフ１１６、１１８、１２０、１２２と呼ばれる制御ブロックに格納される（格納されたビット・パターンは実際にはハッシュ化キーである）。リーフはエージング情報、またはユーザ情報などの追加情報をも含むことができ、それは、目標ブレードまたは目標ポート番号などの転送情報とすることができる。リーフのフォーマットはピココードによって定義され、オブジェクトは内部または外部制御記憶装置に入れられる。

【００４５】ツリーのサーチ・アルゴリズムは、キー１０２を含む入力パラメータに作用し、キーにハッシュ１０４を実行し、直接テーブル（ＤＴ）１０８にアクセスし、パターン・サーチ制御ブロック（ＰＳＣＢ）１１０、１１２、１１４を通してツリーのウォーク・スルーを実行し、リーフ１１６、１１８、１２０、１２２で終了する。各種のツリーはそれ自体のサーチ・アルゴリズムを持ち、異なるルールに従ってツリーのウォークを行わせる。たとえば固定マッチ（ＦＭ）ツリーの場合、データ構造はパトリシア・ツリー・である。リーフが見つかると、このリーフは、入力キー１０２と一致することができる唯一の可能な候補である。「最後の比較」演算で、入力キー１０２はリーフに格納されたパターンと比較される。これにより、リーフが本当に入力キー１０２と一致するかどうかが確認される。このサーチの結果は、リーフが見つかり、かつ一致が発生した場合には成功（ＯＫ）であり、その他の場合は全て失敗（ＫＯ）である。

【００４６】サーチ演算の入力は、次のパラメータを含む。

キー　　サーチまたは挿入／削除の前に、専用ピココード命令を使用して、１７６ビット・キーを構築しなければならない。キー・レジスタは１つしか無い。しかし、ツリー・サーチが始動した後、ピココードによってキー・レジスタを使用して、ＴＳＥ７０でサーチを実行すると同時に次のサーチのためのキーを構築することができる。これは、ＴＳＥ７０がキーをハッシュ化し、結果を内部１９２ビットHashedKeyレジスタ１０６に格納するからである。

17

キー長　この８ビット・レジスタは、キー長から１ビットを引いたものを含む。それは、キーの構築中に、ハードウェアによって自動的に更新される。

LUDefIndex　これは、サーチが行われるツリーの完全な定義を含むルックアップ定義テーブル（LuDefTable）内の８ビット・インデックスである。LUDefTableの内部構造を図１１に示す。

ＴＳＲＮｒ　サーチ結果は、１ビット・ツリー・サーチ結果領域ＴＳＲ０またはＴＳＲ１に格納することができる。これは、ＴＳＲＮｒによって指定される。ＴＳＥがサーチを行っている間、ピココードは他のＴＳＲにアクセスして、前のサーチの結果を分析することができる。

色　色がイネーブルされた（LUDefTableで指定される）ツリーの場合、ハッシュ演算中に、１６ビット色レジスタ１２４の内容がキーに挿入される。

【００４７】ＦＭツリーの場合、入力キーは、図４に示すようにHashedKey１０６にハッシュ化される。利用可能な幾つかの固定アルゴリズムがある。使用されるアルゴリズムは、LUDefTableで指定される。

【００４８】ルックアップ定義テーブルは、ツリー・サーチ・メモリを管理する主要構造である。LUDefTableは内部メモリ構造であり、ツリーを形成するために１２８のエントリを含む。LUDefTableは、ツリーが存在する物理メモリ（たとえばＤＲＡＭ、ＳＲＡＭ、内部ＲＡＭ）、キャッシングがイネーブルされるかどうか、キーおよびリーフのサイズ、ならびに実行すべきサーチ動作の種類を定義するエントリを含む。LUDefTableは、３つの別個のランダム・アクセス・メモリ、つまり一般プロセッサ・ツリー・ハンドラ（ＧＴＨ）によってのみアクセス可能な１つのＲＡＭ、および相互の複製であり全てのピコプロセッサによってアクセス可能な２つのＲＡＭとして実装される。

【００４９】ハッシュ関数１０４の出力は常に１７６ビットの数字であり、元の入力キー１０２とハッシュ関数１０４の出力との間に１対１の対応があるという特性を有する。以下で説明するように、この特性により、直接テーブル１０８の後に始まるツリーの深さは最小になる。

【００５０】図４の例の場合のように、ツリーに色がイネーブルされている場合、１６ビットの色レジスタ１２４が１７６ビットのハッシュ関数出力に挿入され、ファイル結果はHashedKey１０６と呼ばれる１９２ビットの数字となる。挿入は、直接テーブル１０８の直後に行われる。直接テーブル１０８に2^N個のエントリが含まれる場合には、図４に示すように、１６ビットの色値がビット位置Ｎに挿入される。ハッシュ関数の出力は、挿入された色値と一緒に、Hashedkeyレジスタ１０６に格納される。ツリーに対して色がディセーブルされている場合、１７６ビットのハッシュ関数が変更されずに使用され、ハッシュ出力に１６個のゼロが付加され、１９２ビットの最終HashedKeyが生成される。

18

【００５１】色は、複数の独立ツリーの間で単一の直接テーブル１０８を共用するために使用することができる。たとえば色の１つの用途は、ＭＡＣソース・アドレス（ＳＡ）テーブル内のＶＬＡＮ　ＩＤとすることができる。この場合、入力キー１０２はＭＡＣ　ＳＡとなり、色１２４はＶＬＡＮ　ＩＤとなる（ＶＬＡＮ　ＩＤは１２ビットなので、色の４ビットは使用されず、つまりゼロに設定される）。ハッシュ関数１０４の後、使用されるパターンは４８＋１６＝６４ビットである。色は今やパターンの一部であり、異なるＶＬＡＮのＭＡＣアドレスを区別する。

【００５２】ハッシュ関数１０４は、その出力における大部分のエントロピーが最上位ビットに入るように定義される。Hashedkeyレジスタ１０６の最上位のＮビットは、直接テーブル（ＤＴ）１０８内のインデックスを計算するために使用される。

【００５３】ツリーを実装する第１構造は、直接テーブル（ＤＴ）１０８と呼ばれる。Ｎ個の要素を持つＤＴテーブルの各エントリはキーに対応し、その最初の$\log_2 N$ビットは、二進形でＤＴテーブルのそのエントリのインデックスと同じである。たとえば、１６個のエントリのあるＤＴテーブルにおける５番目のエントリは、最初の３ビットが「０１０１」であるキーに対応する。最初の$\log_2 N$ビットがＤＴのインデックスと同じであるキーに対応するリーフが無い場合、そのエントリは空と印される。これらのビットに一致するリーフが１つしか無い場合には、そのエントリの中にリーフのポインタがある。このポインタは、そのリーフが格納されているメモリのアドレスである。最初のビットが同一であるキーに対応するリーフが複数ある場合には、ＤＴエントリはＰＳＣＢ構造１１０を指し、また次の試験ビット（ＮＢＴ）フィールド１２６をも含む。これらの２つの構造について、以下で説明する。

【００５４】ＤＴテーブル１０８はメモリに実装され、そのサイズ（長さ）および開始点はプログラム可能である。別のプログラム可能な特徴は、直接リーフと呼ばれるものを使用することである。ＤＴエントリでリーフを指し示した場合、その後でリーフを読み出さなければならないが、そうする代りにＤＴエントリの位置にリーフを格納することができる。これは、直接リーフと呼ばれる。これに伴う問題は、言うまでもなく、ＤＴエントリにより多くのメモリを使用することによる速度のトレードオフである。メモリ・サイズ（その幅）はリーフを収容するのに充分でなければならず、かつ、全てのＤＴエントリにリーフが格納されるわけではない。しかし、キーの優れたハッシュ関数の結果、リーフのほとんどを単一のＤＴエントリに付加させることができるので、速度のトレードオフが大きくなることがある。

19

【００５５】要するに、ＤＴエントリは空になることがある。この場合、このＤＴエントリにはリーフが付加されない。ＤＴエントリは、このＤＴエントリに付加された単一のエントリを指し示すことができる。この場合、ＤＴエントリはパターン・サーチ制御ブロック（ＰＳＣＢ）を指し示すことができ、またそのＰＳＣＢの次の試験ビット（ＮＢＴ）を含むこともできる。このＤＴエントリに付加されるリーフが２つ以上ある。最後に、ＤＴエントリは直接リーフを含むことができる。

【００５６】ＰＳＣＢは、ツリーの分岐を表す。好ましい実施形態では、０分岐および１分岐がある。ＰＳＣＢから発する分岐の数は、分岐を指定するために使用されるビット数によって異なる。ｎビットが使用される場合には、そのＰＳＣＢに2^n個の分岐が定義される。各ＰＳＣＢはまた、ビット位置ｐにも関連付けられる。ＰＳＣＢから０分岐を通して到達できる全てのリーフは、パターンの位置ｐに「０」を持ち、１分岐を通して到達できるリーフは位置ｐに「１」を持つ。さらに、ＰＳＣＢから到達することのできる全てのリーフは常に、ビット０・・・ｐ－１が同一のパターンを持つ。つまり、パターンは位置ｐから異なり始める。ＰＳＣＢに関連付けられるビット位置は前ＰＳＣＢまたはＤＴエントリに格納され、ＮＢＴ（つまり次の試験ビット）と呼ばれる。ＰＳＣＢエントリのフォーマットは、ＤＴエントリのフォーマットと同一である。それはランダム・アクセス・メモリに実装される。

【００５７】したがって、ＰＳＣＢはツリーのリーフ・パターンが異なる位置に挿入されるだけである。ＰＳＣＢの数、およびしたがってサーチ性能は、ツリー内のリーフの数にのみ依存し、パターンの長さには依存しないので、これにより効率的なサーチ演算が可能になる。ＰＳＣＢレジスタ・フォーマットを図１２に示す。

【００５８】要するに、ＰＳＣＢエントリは空になることができ、リーフを指すことができ、または別のＰＳＣＢを指すことができ、かつ次の試験ビット（ＮＢＴ）をＰＳＣＢに含むこともできる。ＦＭ　ＰＳＣＢは、以下で詳述する通り、１の幅、および１の高さによって定義される形状を持つ。

【００５９】ＰＳＣＢは、２ビット以上に対応する分岐を表すことができる。この場合、たとえば、２ビットに対応するＰＳＣＢは、４つのＰＳＣＢエントリ、つまり００分岐エントリ、０１分岐エントリ、１０分岐エントリ、および１１分岐エントリを持つことになる。各ツリーは、様々なビット数に対応するＰＳＣＢを持つことができる。この場合、前ＰＳＣＢは、これらのビットが表すビット数のみならず、次のＰＳＣＢに対応するビット数をも持つことになる。

【００６０】実際の実装では、キーは専用キー・レジスタ１０２に挿入される。次いでそれはハッシュ化され１０４、結果はハッシュ化キー・レジスタ１０６に格納される。ハッシュ関数１０４はプログラム可能であり、関数の１つは空ハッシュ関数である（つまり、ハッシュ無し）。ハッシュ化キーの最初のｎビットは、ＤＴテーブル１０８のインデックスとして使用される。１つのプログラム可能な機能は、ＤＴエントリのインデックスに使用されるビットの直後にビット・ベクトルを挿入することである。このビット・ベクトルは「色」値（レジスタ１２４）と呼ばれ、ハッシュ化キーおよび挿入された色値の結果は、ハッシュ化キー・レジスタ１０６内に格納される。

20

【００６１】ＦＭツリーのリーフのフォーマットは、パターンを含む制御情報を含む。パターンは、リーフをツリー内で一意のものとして識別する。リーフはまた、ツリー・サーチを開始したアプリケーションによって必要とされるデータをも含む。リーフに含まれるデータはアプリケーションによって異なり、そのサイズまたはメモリ要件は、そのツリーのLUDefTableエントリによって定義される。図１３は、ＦＭツリーの固定リーフ・フォーマットを示す。

【００６２】ＤＴエントリを処理するステップは次の通りである。

・ＤＴエントリがメモリから読み出される。

・ＤＴエントリが空エントリである場合、これは、最初の「ｎ」ビットがハッシュ化キーと同一であるリーフがツリー内に無いことを意味しているので、サーチは失敗する。

・ＤＴエントリにリーフのポインタがある場合には、ＤＴ１０８からのポインタをリーフのアドレスとして使用して、メモリからリーフが読み出される。リーフはレジスタに格納され、キーと比較される。このステップは、最後の比較と呼ばれる。完全一致が存在する場合、ツリー・サーチは成功する。それ以外では、ツリー・サーチは失敗する。

・ＤＴエントリがＰＳＣＢ１１０のポインタおよびＮＢＴを持つ場合、ＮＢＴは最初に特定のレジスタに格納される。次いでＮＢＴ数は、キー内のＮＢＴ位置のビットを見つけるために使用される。そのビット（０または１）をＰＳＣＢのポインタと共に使用して、正確なＰＳＣＢエントリが抽出される。つまり、そのビットはポインタの終わりに付加され、ＰＳＣＢのメモリ内の完全なアドレスを提供する。ＰＳＣＢは読み出され、特定のレジスタに格納される。次いでハードウェアはＰＳＣＢエントリを処理する。この時点で、アルゴリズムはツリーを下方に辿るウォークを開始する。

【００６３】ＰＳＣＢエントリを処理するステップは次の通りである。

・ＰＳＣＢエントリが空エントリである場合、これは、最初のＮＢＴビットがキーと同一であるリーフがツリー内に無いことを意味するので、サーチは失敗する。

・ＰＳＣＢにリーフのポインタがある場合には、ＰＳＣ

21

Ｂからのポインタをリーフのアドレスと使用して、メモリからリーフが読み出される。リーフはレジスタに格納され、キーと比較される。このステップは、最後の比較と呼ばれる。完全一致が存在する場合、ツリー・サーチは成功する。それ以外では、ツリー・サーチは失敗する。

・ＰＳＣＢがＰＳＣＢのポインタおよびＮＢＴを持つ場合、ＮＢＴは最初に特定のレジスタに格納され、これは現在のＮＢＴになる。次いでこのＮＢＴ数は、キー内のＮＢＴ位置のビットを見つけるために使用される。そのビット（０または１）をＰＳＣＢのポインタと共に使用して、正確な次のＰＳＣＢエントリが抽出される。そのビットはポインタの終わりに付加され、ＰＳＣＢのメモリ内の完全なアドレスを提供する。ＰＳＣＢは読み出され、特定のレジスタに格納される。次いでハードウェアは、ＰＳＣＢエントリのこの処理を繰り返す。

【００６４】ツリー・ウォーク中、リーフの全てのビットが試験されるのではなく、ＰＳＣＢ（ツリーの分岐）があるビットだけが試験される。したがって、リーフが見つかったら、リーフのパターンをキーと比較して、全てのビットが一致することを確認しなければならない。これが、このアルゴリズムの演算の最後の比較の理由である。サーチの成否は、完了フラッグと共にＯＫ／ＫＯフラッグによって印される。完了フラッグがトリガされると、このＦＭツリー・サーチ・エンジンを使用するプログラムまたはハードウェアは、ＯＫ／ＫＯフラッグを調べることができる。

【００６５】「プログラム可能」と記載されたものは全て、そのツリーに対応する特定のレジスタ値に設定することができる。エンジンがＮ個のツリーをサポートする必要がある場合には、これらの値がＮ個、レジスタ・アレイに入れられる。このレジスタには、プログラム可能な値、つまり使用するハッシュ関数、ＤＴテーブルの開始、そのサイズ等が符号化される。

【００６６】ハードウェアの１つの能力として、キーの自動挿入（ハードウェア挿入）がある。（ハッシュ化された）キーのサーチを進めながら、不一致（ＫＯ）があった場合、ハードウェアを使用して進行中にＰＳＣＢを形成することによって、そのポイントにリーフを自動的に挿入することができる。この場合、完全一致ツリーの概念をキャッシュとして使用することができる。

【００６７】サーチは、直接テーブル１０８へのアクセスにより開始される。つまり、ＤＴエントリは直接テーブル１０８から読み出される。ＤＴエントリを読み出すために使用されるアドレスは、HashedKeyの最高位Ｎビットからだけでなく、ルックアップ定義テーブル（LUDefTable）で定義されるツリー特性に基づいても計算される。ＤＴエントリは、ツリーのルートとみることができる。実際のツリー・データ構造は、ツリーの型に依存する。パトリシア・ツリー・データ構造はＦＭツリーに使用され、パトリシア・ツリーの延長はＬＰＭおよびＳＭＴツリーに使用される。

22

【００６８】８つのエントリのあるＤＴ１０８の使用例を図６に示す。サーチ時間つまりアクセスしなければならないＰＳＣＢの数は、ＤＴ１０８を使用することによって減少できることが分かる。したがって、ＤＴサイズを増加することによって、メモリの使用量とサーチ性能との間でトレードオフを行うことができる。

【００６９】性能上の理由から、ＤＴエントリを読み出して、そこにリーフのポインタが含まれることを見出すだけでは不十分であり、その後でリーフ自体を読み出さなければならない。この状況は、ＤＴエントリ当たりの単一リーフ・エントリが多数存在するＦＭツリーの場合、非常にしばしば発生する。直接リーフの概念は、より多くのメモリの使用量とより優れた性能との間のトレードオフを可能にする。

【００７０】ツリーは直接リーフをイネーブルすることができ、これはルックアップ定義テーブル（LUDefTable）で指定される。直接リーフをイネーブルしたツリーとディセーブルしたツリーとの相違を、図７に示す。直接リーフがイネーブルされ、ＤＴエントリが単一リーフを含む場合、このリーフ１３０はＤＴエントリ自体に直接格納される。それ以外の場合、ＤＴエントリはリーフのポインタを含む。

【００７１】形状設定はツリー・サーチ・メモリ（ＴＳＭ）の特徴であり、リーフまたはＰＳＣＢのようなオブジェクトをＴＳＭにどのように格納するかを指定するために使用される。形状は幅および高さパラメータによって定義される。オブジェクトの高さは、オブジェクトが格納される連続アドレス場所の数を示す。オブジェクトの幅は、オブジェクトが格納される連続バンクの数を示す。幅および高さについて、ハードウェアは自動的に適切な数の場所を読み出す。ピココードの観点から、オブジェクトはアクセスの最小単位である。幅は、ＳＲＡＭに格納されたオブジェクトの場合、常に１としなければならない。幅は、ＤＲＡＭのオブジェクトの場合、２以上にすることができる。単一メモリ場所内に収まる充分に小さいオブジェクトは、１の高さおよび１の幅を持つように定義される。直接リーフをディセーブルしたＤＴエントリの形状は常に（Ｗ＝１、Ｈ＝１）である。ＤＴエントリがダイナミック・ランダム・アクセス・メモリ（ＤＲＡＭ）に格納される場合、それはちょうど６４ビットを占有する。直接リーフをイネーブルしたＤＴエントリの形状は、リーフの形状に等しく、それはLUDefTableに指定される。一般的に、これは、ＤＴ１０８によって使用されるメモリを増加させる。それはまた、ＤＴエントリのアドレス計算に対するリーフの形状の影響を生じさせる。

【００７２】ＤＴエントリが読み出された後、ＤＴエントリが直接リーフを含まず、空でもないと仮定して、Ｄ

23

Ｔエントリから始まるツリーをウォークすることによって、サーチが続行される。ツリー・ウォークは、リーフに達するまで幾つかのＰＳＣＢ（パターン・サーチ制御ブロック）を通過することができる。

【００７３】ＦＭツリーのサーチ中にＰＳＣＢに遭遇すると、ツリー・サーチ・エンジン・ハードウェア７０は、HashedKeyのビットｐの値によって、０分岐または１分岐にツリー・ウォークを続行する。

【００７４】ツリー・ウォーク中、HashedKeyの全てのビットが試験されるのではなく、ＰＳＣＢがあるビットだけが試験される。したがって、リーフが見つかったら、リーフのパターンをまだHashedKeyと比較して、全てのビットが一致することを確認しなければならない。リーフに格納されているのはHashedKeyであって、元の入力キーではないことに注意されたい。ＦＭリーフが見つかったら、次の操作が実行される。

ステップ１：リーフ・パターンがHashedKeyと比較される。一致が発生すると、操作はステップ２に進む。そうでない場合、リーフが別のリーフへのチェーン・ポインタを含むならば、このリーフが読み出され、パターンが再びHashedKeyと比較される。一致せず、かつＮＬＡフィールドが無ければ、サーチは失敗（ＫＯ）で終了する。

ステップ２：ベクトル・マスクがイネーブルされている場合、VectorIndex数を持つビットがリーフのベクトル・マスクから読み出される。このビットは、サーチ結果の一部として返される。サーチは成功（ＯＫ）で終了する。

【００７５】図１０は、本発明の完全一致サーチ・アルゴリズムの処理論理を示す。処理は、論理ブロック１０００で入力キーの読出しから始まる。次いで入力キーに、論理ブロック１００２に示すようにハッシュ関数が行われる。入力キーでハッシュ化を行なってハッシュ化キーを形成することは、任意選択である。ハッシュ関数は、ハッシュ化キーの最左端ビット、つまり直接テーブルをアドレス指定するために使用されるビットでエントロピーが最高となるように、選択される。ハッシュ関数は可逆である。つまり、ハッシュ化キーを入力キーに変換できる逆方向ハッシュ関数が存在する。次に、論理ブロック１００４で、直接テーブルが読み出される。ハッシュ化キーの上位Ｎビット（それによりＮは構成可能である）は、直接テーブルのインデックスとして使用される。読み出されたエントリが空である場合、サーチはＫＯ（一致が見つからない）を返す。これは、終了ブロック１００６によって示される。決定ブロック１００８に示すように、エントリがリーフを指し示すか否かに関して決定が行われる。ＤＴエントリがリーフを指し示す場合には、論理ブロック１０１０に示すように、そのリーフが読み出される。そうでない場合には、ＤＴエントリはＰＳＣＢを指し示す。この場合、論理ブロック１０１２に示すように、ＰＳＣＢの適切な部分が読み出される。完全一致サーチの場合、ＰＳＣＢは２つのエントリ、つまり０部分と１部分を含む。前ＰＳＣＢ（またはＤＴエントリ）は、ビット番号（ＮＢＴ：試験する次のビット）を含む。ＮＢＴは、どのＰＳＣＢエントリを使用するかを選択するハッシュ化キー内のビット（つまり０または１）を選択する。ＰＳＣＢエントリは、リーフのポインタまたは別のＰＳＣＢのポインタのいずれかを含む。次いで、処理は決定ブロック１００８にループバックする。決定ブロック１００８でリーフが見つかり、論理ブロック１０１０で読み出されると、論理ブロック１０１４で示すように、リーフに格納されたパターンが１ビットずつハッシュ化キーと比較される。決定ブロック１０１６に示すように、全てのビットが一致すると、サーチは、終了ブロック１０１８に示すように、ＯＫ（一致の成功）を返す。次いで、リーフの内容はアプリケーションに渡される。一致しない場合、終了ブロック１０２０に示すように、サーチはＫＯ（失敗）を返す。この処理論理の延長として、ハッシュ化キーからのｂビットでどのエントリをＰＳＣＢから読み出すかを選択するように、ＰＳＣＢは2^b個のエントリで構成することができる。これにより、より多くのメモリ使用量を犠牲にして、性能が増強される。

24

【００７６】ツリーのサーチ性能を増強するために、キャッシュを使用することができる。キャッシュの使用は、LUDefTableでツリー毎にイネーブルすることができる。サーチ中、ツリー・サーチ・エンジン７０は最初にキャッシュを検査して、HashedKeyと一致するリーフが存在するかどうかを決定する。そのようなリーフが見つかると、それが返され、それ以上のサーチは必要ない。そのようなリーフが見つからなければ、通常のサーチが開始される。

【００７７】ツリー・サーチ・エンジン・ハードウェア７０にとって、キャッシュ・ルックアップは通常のサーチと全く同一である。したがって、入力キーはHashedKeyにハッシュ化され、直接テーブル１０８のアクセスが実行される。直接テーブル１０８はキャッシュとして作動する。キャッシュ・サーチがＯＫ（成功）を返すと、サーチは終了する。そうでなければ、ツリー・サーチ・エンジン７０は、ハッシュ演算が行われないことを除いて、全ツリーの第２サーチを開始する。HashedKeyレジスタ１０６の内容は再使用される。

【００７８】キャッシュ・サーチを使用する場合、それはLUDefTableで指定することができる。キャッシュ・サーチがLUDefTableエントリＩを使用し、サーチがＫＯ（失敗）で終了すると、LUDefTableエントリＩ＋１を使用して、別のサーチが自動的に開始される。原則的に、これにより複数のサーチを連鎖的に行うことができるが、LUDefTableのエントリＩ＋１の下にツリー全体を格納することが推薦される。

25

【００７９】ツリー・サーチ・エンジン７０は、ＦＭツリー、ＬＰＭツリー、およびＳＭＴツリーでのハードウェア・サーチ作業を行う。三種類全ての場合に、ツリーを初期化し、維持するために変化する量のソフトウェアが必要になる。ＦＭツリーおよびＬＰＭツリーのみが、制御点プロセッサ３４の介在無く、リーフの挿入および除去を実行できる能力を持つ。この機能の使用により、スケーラブルな構成が可能になり、かつ、必要な場合には、依然としてＣＰ３４がリーフを挿入または除去することができる柔軟性を備えている。

【００８０】ＦＭツリーは、テーブルで固定サイズ・パターンを効率的にサーチするための機構を提供する。これの一例は、レイヤ２イーサネット・ユニキャストＭＡＣテーブルである。イーサネット・ユニキャストＭＡＣアドレスは固定長の６バイトであり、厳密に一致しなければならず、さもなければ宛先は不明である。

【００８１】ＦＭツリーは、ハッシング関数から著しい利益が得られるので、最も性能の良いツリーである。ツリー・サーチ・エンジンは、衝突率が非常に低い複数の固定ハッシング関数を提供する。ＤＴ１０８が充分に大きいと仮定すると、複数のリーフが単一のＤＴエントリに関連付けられる確率は非常に低い。これは１＋イプシロン・ルールであり、ここでイプシロンはＤＴエントリにおける衝突数を表す。１つのリーフを持つＤＴエントリは、イプシロン＝０である。したがって、ハッシング関数を用いて、ＦＭツリーを使用すると、イプシロンの値は非常に小さくなるはずである。

【００８２】ＦＭツリーのＤＴエントリの構造は、図８に見ることができる。各ＤＴエントリは３６ビット幅であり、次のフォーマットの１つを含む。

・空のＤＴエントリ。このＤＴエントリに関連付けられるリーフは無い。

・次のＰＳＣＢのポインタ。ＤＴエントリは、ＰＳＣＢのポインタを含む。次のＰＳＣＢアドレス（ＮＰＡ）フィールドおよび次の試験ビット（ＮＢＴ）フィールドが有効である。

・リーフのポインタ。ＤＴエントリに関連付けられる単一のリーフがある。リーフ制御ブロック・アドレス（ＬＣＢＡ）は、このリーフのポインタを含む。

・直接リーフ。ＤＴエントリに関連付けられる単一のリーフがあり、リーフはＤＴエントリ自体に格納される。リーフの第１フィールドはＮＬＡロープでなければならず、これは直接リーフがループをイネーブルしなければならないことを暗示する。ループとは、ツリー内のリーフをひとつに連結するために使用される循環連結リストである。ピココードは「ロープをウォーク」して、ロープ内の全てのリーフを順次検査することができる。ＮＬＡの最初の２ビットは、自動的に「直接」と符号化するように、「１０」を表すように予約されることに留意されたい。直接リーフは、これがLUDefTableでイネーブルされた場合に、所与のツリーのためにだけ使用される。

26

・ＦＭ　ＰＳＣＢは、２つのＰＳＣＢラインで構成され、各ＰＳＣＢラインが図８に示す２つのフォーマットのうちの１つを持つことができることを除いては、ＦＭ　ＤＴエントリと同一の構造を持つ。２つのＰＳＣＢラインはメモリ内で連続的に割り当てられ、ツリーをウォークするための分岐として使用される。次の試験ビット（ＮＢＴ）フィールドは、ＰＳＣＢをウォークするためのビット比較として使用するキーのオフセットを意味し、２つのＰＳＣＢラインのどちらを使用するかを表す。

【００８３】７ビット値がツリーに格納される場合のＦＭツリーのサーチの一例を、図９に見ることができる。この例は、キーの最上位３ビット（ＭＳＢ）をＦＭ　ＤＴ１０８へのハッシュとして使用することによって単純化される。このツリーに格納された５つのリーフ・エントリ（Ｌ０－Ｌ４）がある。

【００８４】第１例として、１１１００１１のバイナリ入力キーを想定する。最初の３ビット「１１１」はＤＴエントリ７に指標を付け、ここでリーフＬ０を指すＬＣＢＡが存在する。リーフＬ０はＴＳＥ７０によって読み出され、Ｌ０のパターンが入力パターンと比較される。この例では、厳密な一致が発生し、ＴＳＥはＯＫ（成功）を返す。

【００８５】今、１００１１１０の入力パターンを想定する。ＤＴエントリ４は、ＮＢＴフィールドが３のＰＳＣＢ０のポインタを含む。これは、キーの第４ビット「１」（ビット０はＭＳＢつまり最上位ビット）がツリーのどの分岐を選ぶかを決定することを意味する。第４ビットは「１」なので、ＰＳＣＢ０の下半分が使用される。それが「０」であった場合、ＰＳＣＢ０の上半分が使用される。各ＰＳＣＢは基本的にＰＳＣＢラインの２要素配列であり、ここで「０」のＮＢＴ値は第１要素を指標付けし、「１」のＮＢＴ値は第２要素を指標付けする。したがって、ＰＳＣＢ０のＰＳＣＢライン１は、６のＮＢＴおよびＰＳＣＢ２を指す次のＰＳＣＢアドレス（ＮＰＡ）を含むので、サーチが続行される。ＮＢＴが７であり、入力パターンのビット７が「０」である場合、Ｌ３のポインタを含むＰＳＣＢ２の上半分が使用される。リーフＬ３を読出し、Ｌ３のパターンと入力パターンの完全比較作業を実行すると、ＯＫ（成功）が返される。

【００８６】入力パターン１００１１００のサーチは、前の例の場合と全く同じツリー内の経路を辿るが、最後の比較作業は一致しないので、サーチはＫＯ（失敗）を返す。

【００８７】本発明は、ハードウェア、ソフトウェア、または２つの組合せで実現することができる。ここで説明する方法を実行するために適応させた任意の種類のコンピュータ・システムまたはその他の装置が適合する。

ハードウェアとソフトウェアの一般的な組合せは、ロードされ実行したときに、ここで説明する方法を実行するようにコンピュータ・システムを制御する、汎用コンピュータ・システムとすることができる。本発明はまた、ここで説明した方法の実現を可能にする全ての機能を含み、コンピュータ・システムにロードされたときにこれらの方法を実行することができる、コンピュータ・プログラム製品に埋め込むこともできる。

【００８８】本発明の文脈におけるコンピュータ・プログラム命令またはコンピュータ・プログラムとは、情報処理能力を有するシステムに特定の関数をすぐに、またはａ）別の言語、コードまたは表記法への変換、およびｂ）異なるマテリアル形式での複製、のいずれかまたは両方が行われた後で、実行させるように意図された１組の命令のいずれかの言語、コード（つまりピココード命令）、または表記法による表現を意味する。

【００８９】当業者は、本発明の精神および範囲から逸脱することなく、本発明の好ましい実施形態に多くの変更が可能であることを理解されるであろう。さらに、他の機能を相応じて使用することなく、本発明の特徴の幾つかを使用することが可能である。したがって、本発明の範囲は請求の範囲によってのみ画定されるので、好ましい実施形態の上記の説明は、本発明の原理を解説することを目的として、本発明を制限することなく、提供するものである。

【００９０】まとめとして、本発明の構成に関して以下の事項を開示する。

【００９１】（１）入力キーをサーチ文字列として読み出す動作と、ハッシュ関数を使用して前記入力キーをハッシュ化してハッシュ化キーを生成する動作と、前記ハッシュ化キーの最上位Ｎビットを、各々の非空エントリがサーチ・ツリーの次の分岐またはリーフのポインタを含むサーチ・ツリーの複数のルート・ノードを表すテーブルのインデックスとして使用する動作と、非空テーブル・エントリのポインタが、対応するサーチ・ツリーのリーフまたは次の分岐を指し示すかどうかを決定する動作と、ポインタが対応するサーチ・ツリーのリーフを指し示さない場合、次の分岐内容を読み出す動作と、対応するサーチ・ツリーのリーフに達した場合、リーフ内容を読み出し、リーフのパターンを前記ハッシュ化キーと比較して、前記リーフ・パターンが前記ハッシュ化キーと一致するかどうかを決定する動作と、前記リーフ・パターンが前記ハッシュ化キーと一致する場合、要求するアプリケーションに見つかったリーフの内容を返す動作とを含む、コンピュータ処理装置によって可変長サーチ・キーの完全一致を決定するための方法。

（２）前記サーチ・ツリーの複数のルート・ノードを表すテーブルが2^N個のエントリを含む、上記（１）に記載の完全一致を決定するための方法。

（３）前記コンピュータ処理装置がネットワーク・プロセッサである、上記（１）に記載の完全一致を決定するための方法。

（４）前記対応するサーチ・ツリーの次の分岐の内容が別の次の分岐を指し示す、上記（１）に記載の完全一致を決定するための方法。

（５）前記次の分岐の内容が対応するサーチ・ツリーのリーフを指し示す、上記（１）に記載の完全一致を決定するための方法。

（６）前記リーフ・パターンが前記ハッシュ化キーと一致せず、かつ別のリーフのポインタを含まない場合、一致見つからずの標識を返すことをさらに含む、上記（１）に記載の完全一致を決定するための方法。

（７）テーブルのインデックスが空エントリのインデックスである場合、一致見つからずの標識を返すことをさらに含む、上記（１）に記載の完全一致を決定するための方法。

（８）色レジスタの内容を前記ハッシュ化キーに付加して最終ハッシュ化キーを提供することをさらに含む、上記（１）に記載の完全一致を決定するための方法。

（９）ゼロの文字列を前記ハッシュ化キーに付加して最終ハッシュ化キーを提供することをさらに含む、上記（１）に記載の完全一致を決定するための方法。

（１０）前記次の分岐のビット数が前記ハッシュ化キーの長さを超える場合、完全一致のサーチを終了する動作をさらに含む、上記（１）に記載の完全一致を決定するための方法。

（１１）前記入力キーに対して使用される前記ハッシュ関数が、前記ハッシュ化キーを前記入力キーに変換できる可逆ハッシュ関数である、上記（１）に記載の完全一致を決定するための方法。

（１２）前記リーフが別のリーフへのチェーン・ポインタを含む場合、別のリーフに格納されたパターンを読み出し、前記パターンを前記ハッシュ化キーと比較する動作と、前記格納されたパターンが前記ハッシュ化キーと一致せず、かつ前記チェーンの次のリーフのポインタを含まない場合、一致見つからずの標識を返す動作とをさらに含む、上記（１）に記載の完全一致を決定するための方法。

（１３）前記リーフが別のリーフのチェーン・ポインタを含む場合、別のリーフに格納されたパターンを読み出し、前記パターンを前記ハッシュ化キーと比較する動作と、前記格納されたパターンが前記ハッシュ化キーと一致する場合、一致発見の標識を返す動作とをさらに含む、上記（１）に記載の完全一致を決定するための方法。

（１４）サーチすべきパターンまたはキーと、サーチ・ツリーの第１アドレス位置を格納する直接テーブルと、各々がサーチ・ツリーの分岐を表す複数のパターン・サーチ制御ブロックと、各リーフがサーチの結果のためのアドレス位置である複数のリーフとを含む、可変長サー

チ・キーの完全一致を見つけるための複数のデータ構造を含むコンピュータ読み出し可能な媒体。

（15）ツリー・サーチ・メモリを管理するルックアップ定義テーブルをさらに含む、上記（14）に記載の完全一致を見つけるための複数のデータ構造を含むコンピュータ読み出し可能な媒体。

（16）前記ルックアップ定義テーブルは、ツリーが存在する物理的メモリ、キーおよびリーフのサイズ、および実行すべきサーチの種類を定義するエントリを含む、上記（15）に記載の完全一致を見つけるための複数のデータ構造を含むコンピュータ読み出し可能な媒体。

（17）前記ルックアップ定義テーブルが複数のメモリに実装される、上記（14）に記載の完全一致を見つけるための複数のデータ構造を含むコンピュータ読み出し可能な媒体。

（18）直接テーブル・エントリのフォーマットがサーチ制御ブロック、次のパターン・サーチ制御ブロックを指し示す次のパターン・アドレス、リーフまたは結果を指し示すリーフ制御ブロック・アドレス、次の試験ビット、および直接リーフのうちの少なくとも1つを含む、上記（14）に記載の完全一致を見つけるための複数のデータ構造を含むコンピュータ読み出し可能な媒体。

（19）パターン・サーチ制御ブロックのフォーマットがサーチ制御ブロック、次のパターン・サーチ制御ブロックを指し示す次のパターン・アドレス、リーフまたは結果を指し示すリーフ制御ブロック・アドレス、および次の試験ビットのうちの少なくとも1つを含む、上記（14）に記載の完全一致を見つけるための複数のデータ構造を含むコンピュータ読み出し可能な媒体。

（20）リーフ・データ構造が1つのリーフ連鎖ポインタ、プレフィックス長、サーチ・キーと比較すべきパターン、および可変ユーザ・データのうちの少なくとも1つを含む、上記（14）に記載の完全一致を見つけるための複数のデータ構造を含むコンピュータ読み出し可能な媒体。

（21）前記直接リーフは直接テーブル・エントリに直接格納され、かつサーチ制御ブロックおよびサーチ・キーと比較されるパターンを含む、上記（18）に記載の完全一致を見つけるための複数のデータ構造を含むコンピュータ読み出し可能な媒体。

（22）パターン・サーチ制御ブロックが、サーチ・ツリーのリーフ・パターンが異なる位置に挿入される、上記（14）に記載の完全一致を見つけるための複数のデータ構造を含むコンピュータ読み出し可能な媒体。

（23）パターン・サーチ制御ブロックは1の幅および1の高さによって定義される形状を持ち、かつ少なくとも36ビットのライン長を有するメモリに格納される、上記（14）に記載の完全一致を見つけるための複数のデータ構造を含むコンピュータ読み出し可能な媒体。

（24）フレーム処理を行う複数のプロトコル・プロセッサおよび内部制御点プロセッサを含む埋込み式プロセッサ複合体と、各プロトコル・プロセッサにアクセスでき、高速パターン・サーチ、データ操作、およびフレーム・パージングをもたらす複数のハードウェア・アクセラレータ・コプロセッサと、少なくとも1つのサーチ・ツリーを表す複数のデータ構造であって、直接テーブルとパターン・サーチ制御ブロックとリーフとを含む前記データ構造を格納する複数のプログラム可能なメモリ装置と、各プロトコル・プロセッサの複数のメモリ装置へのアクセスを制御する制御メモリ・アービタとを含む、可変長サーチ・キーの完全一致を決定するために半導体基板上に組み立てられた装置。

（25）プロトコル・プロセッサの実行と並行して作動して、メモリの読み書きおよびメモリ範囲検査を含むツリー・サーチ命令を実行するツリー・サーチ・エンジンをさらに含む、上記（24）に記載の完全一致を決定するために半導体基板上に組み立てられた装置。

（26）前記複数のメモリ装置は内部スタティック・ランダム・アクセス・メモリ、外部スタティック・ランダム・アクセス・メモリ、および外部ダイナミック・ランダム・アクセス・メモリの少なくとも1つをさらに含む、上記（24）に記載の完全一致を決定するために半導体基板上に組み立てられた装置。

（27）前記制御メモリ・アービタが、複数のプロトコル・プロセッサと複数のメモリ装置との間でメモリ・サイクルを割り当てることによって、制御メモリ動作を管理する、上記（24）に記載の完全一致を決定するために半導体基板上に組み立てられた装置。

（28）各プロトコル・プロセッサは一次データ・バッファ、スクラッチ・パッド・データ・バッファ、およびデータ格納動作のための制御レジスタを含む、上記（24）に記載の完全一致を決定するために半導体基板上に組み立てられた装置。

（29）サーチ・キーに対して幾何学的ハッシュ関数を実行するハッシュ・ボックス・コンポーネントをさらに含む、上記（24）に記載の完全一致を決定するために半導体基板上に組み立てられた装置。

（30）プログラム可能なサーチ・キー・レジスタおよびプログラム可能なハッシュ化キー・レジスタをさらに含む、上記（24）に記載の完全一致を決定するために半導体基板上に組み立てられた装置。

（31）複数の独立サーチ・ツリーの間で単一のデーブル・データ構造を共用することを可能にするプログラム可能な色キー・レジスタをさらに含む、上記（30）に記載の完全一致を決定するために半導体基板上に組み立てられた装置。

（32）色レジスタがイネーブルされている場合、その内容をハッシュ出力に付加して最終ハッシュ化キーを生成する、上記（31）に記載の完全一致を決定するために半導体基板上に組み立てられた装置。

31

（３３）色レジスタがイネーブルされていない場合、同等の数のゼロをハッシュ出力に付加して最終ハッシュ化キーを生成する、上記（３１）に記載の完全一致を決定するために半導体基板上に組み立てられた装置。

（３４）入力キーをサーチ文字列として読み出すプログラム命令と、ハッシュ化キーを入力キーに変換することができる可逆ハッシュ関数を使用して前記入力キーをハッシュ化するプログラム命令と、前記ハッシュ化キーの最上位Ｎビットを、各非空エントリがサーチ・ツリーの次の分岐またはリーフへのポインタを含むサーチ・ツリーの複数のルート・ノードを表すテーブルのインデックスとして使用するプログラム命令と、非空テーブル・エントリのポインタが対応するサーチ・ツリーのリーフまたは次の分岐を指し示すかどうかを決定するプログラム命令と、ポインタが対応するサーチ・ツリーのリーフを指し示さない場合、次の分岐の内容を読み出すプログラム命令と、対応するサーチ・ツリーのリーフに達したときにリーフ内容を読み出し、リーフのパターンをハッシュ化キーと比較して、リーフ・パターンがハッシュ化キーと一致するか否かを決定するプログラム命令と、リーフ・パターンがハッシュ化キーと一致する場合、要求するアプリケーションに見つかったリーフの内容を返すプログラム命令とを含む、可変長サーチ・キーの完全一致を決定するためのコンピュータ・プログラム製品を含むコンピュータ読み出し可能な媒体。

（３５）サーチ・ツリーの複数のルート・ノードを表すテーブルが2^N個のエントリを含む、上記（３４）に記載の完全一致を決定するためのコンピュータ・プログラム製品。

（３６）前記コンピュータ処理装置がネットワーク・プロセッサである、上記（３４）に記載の完全一致を決定するためのコンピュータ・プログラム製品。

（３７）前記対応するサーチ・ツリーの次の分岐の内容が別の次の分岐を指し示す、上記（３４）に記載の完全一致を決定するためのコンピュータ・プログラム製品。

（３８）前記次の分岐の内容が対応するサーチ・ツリーのリーフを指し示す、上記（３４）に記載の完全一致を決定するためのコンピュータ・プログラム製品。

（３９）リーフ・パターンがハッシュ化キーと一致せず、かつ別のリーフのポインタを含まない場合、一致見つからずの標識を返すプログラム命令をさらに含む、上記（３４）に記載の完全一致を決定するためのコンピュータ・プログラム製品。

（４０）テーブルのインデックスが空エントリのインデックスである場合、一致見つからずの標識を返すプログラム命令をさらに含む、上記（３４）に記載の完全一致を決定するためのコンピュータ・プログラム製品。

（４１）色レジスタの内容をハッシュ化キーに付加して最終ハッシュ化キーを提供するプログラム命令をさらに含む、上記（３４）に記載の完全一致を決定するための

32

コンピュータ・プログラム製品。

（４２）文字列またはゼロをハッシュ化キーに付加して最終ハッシュ化キーを提供するプログラム命令をさらに含む、上記（３４）に記載の完全一致を決定するためのコンピュータ・プログラム製品。

（４３）次の分岐のビット数がハッシュ化キーの長さを超える場合、完全一致のサーチを終了するプログラム命令をさらに含む、上記（３４）に記載の完全一致を決定するためのコンピュータ・プログラム製品。

（４４）リーフが別のリーフのチェーン・ポインタを含む場合、別のリーフに格納されたパターンを読み出し、前記パターンをハッシュ化キーと比較するプログラム命令と、格納されたパターンがハッシュ化キーと一致せず、かつチェーンの次のリーフのポインタを含まない場合、一致見つからずの標識を返すプログラム命令とをさらに含む、上記（３４）に記載の完全一致を決定するためのコンピュータ・プログラム製品。

（４５）リーフが別のリーフのチェーン・ポインタを含む場合、別のリーフに格納されたパターンを読み出し、前記パターンをハッシュ化キーと比較するプログラム命令と、格納されたパターンがハッシュ化キーと一致する場合、一致発見の標識を返すプログラム命令とをさらに含む、上記（３４）に記載の完全一致を決定するためのコンピュータ・プログラム製品。

【図面の簡単な説明】

【図１】本発明の好ましい実施形態によるネットワーク・プロセッサの例示的アーキテクチャを示すブロック図である。

【図２】本発明の好ましい実施形態による埋込み型プロセッサ複合体の例示的実施形態を示すブロック図である。

【図３】本発明の好ましい実施形態による例示的プロトコル・プロセッサ構造を示すブロック図である。

【図４】本発明の好ましい実施形態による例示的イングレスおよびエグレス・フレームの流れを示すブロック図である。

【図５】本発明の好ましい実施形態による完全マッチ・サーチ・アルゴリズムのためのツリー・データ構造を示す略図である。

【図６】本発明の好ましい実施形態による直接テーブルを使用することの例示的データ構造への効果を示す略図である。

【図７】本発明の好ましい実施形態による直接リーブを使用可能にすることの例示的データ構造への効果を示す略図である。

【図８】本発明の好ましい実施形態による完全マッチ・サーチ・ツリーのＤＴエントリおよびパターン・サーチ制御ブロック（ＰＳＣＢ）ライン・フォーマットの例示的構造を示す表である。

【図９】本発明の好ましい実施形態による完全マッチ・

サーチを使用するサーチの例を示す略図である。

【図１０】本発明の好ましい実施形態による完全マッチ（ＦＭ）サーチ・アルゴリズムの処理論理を示す図である。

【図１１】本発明の好ましい実施形態による例示的参照定義テーブルの内部構造を示す表である。

【図１２】ＰＳＣＢレジスタの内部フォーマットを示す表である。

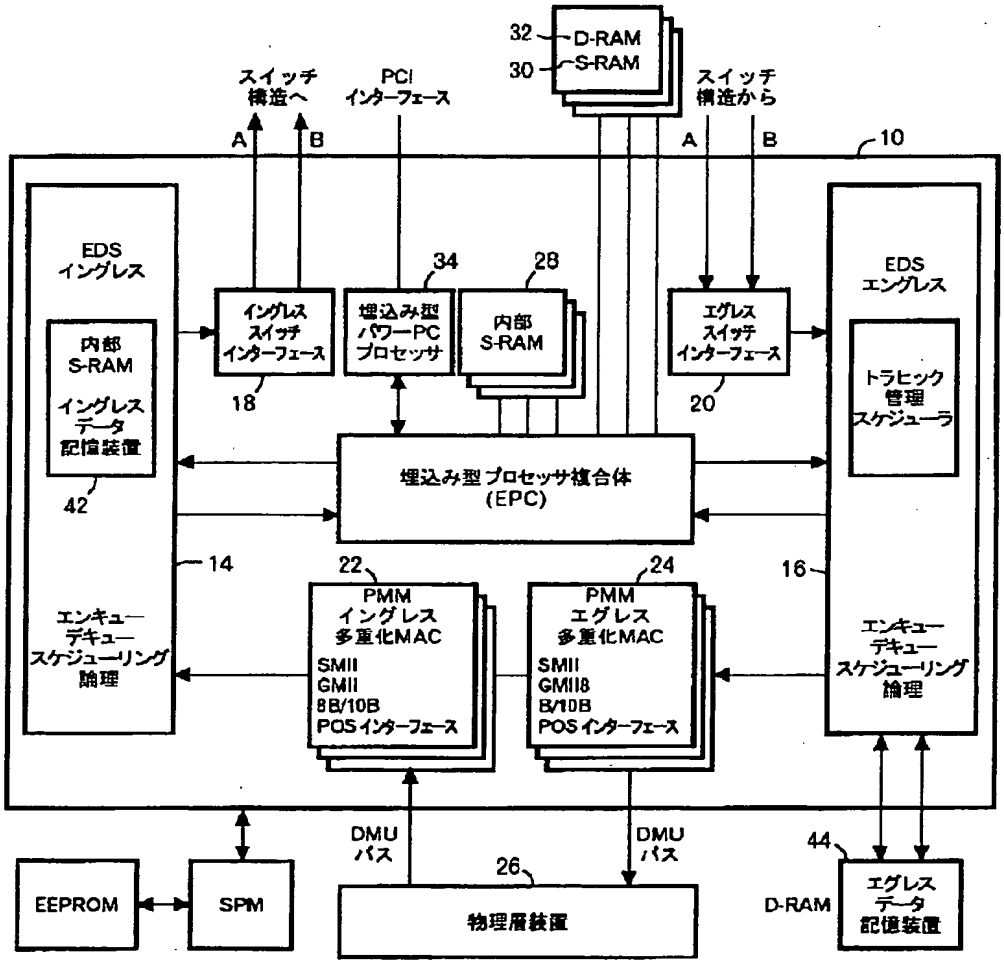
【図１３】ＦＭツリーの固定リーフ・フォーマットを示す表である。

【図１４】本発明の好ましい実施形態によるツリー・サーチ・エンジンの例示的アーキテクチャを示すブロック図である。

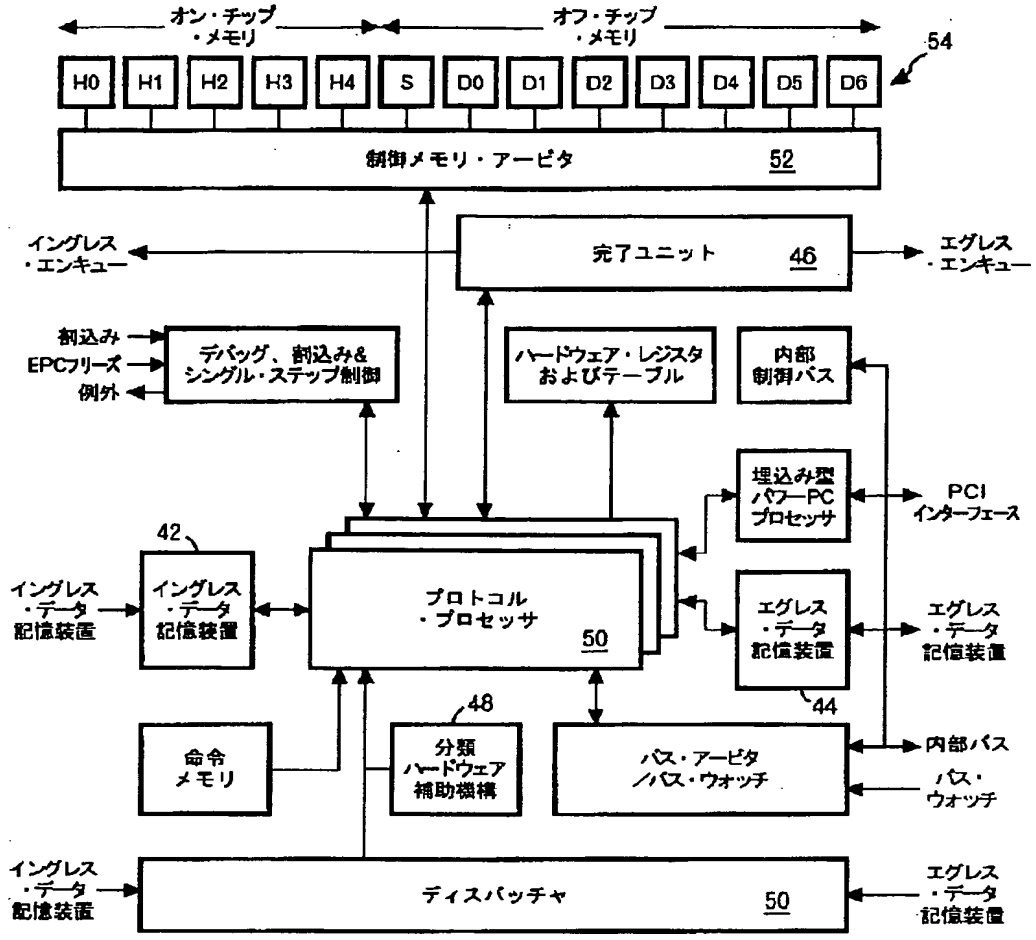
【符号の説明】

１０　ネットワーク・プロセッサ
１２　埋込み型プロセッサ複合体
１４　エンキュー・デキュー・スケジューリング論理
１８　イングレス・スイッチ・インタフェース
２０　エグレス・スイッチ・インタフェース
２８　スタティック・ランダム・アクセス・メモリ（ＳＲＡＭ）
３０　ゼロ・バス・ターンアラウンドＳＲＡＭ
３２　ダイナミック・ランダム・アクセス・メモリ（ＤＲＡＭ）
３４　接続点プロセッサ

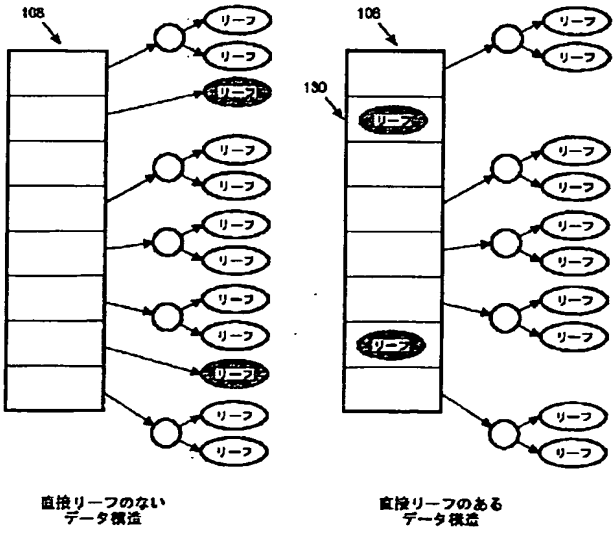
【図１】



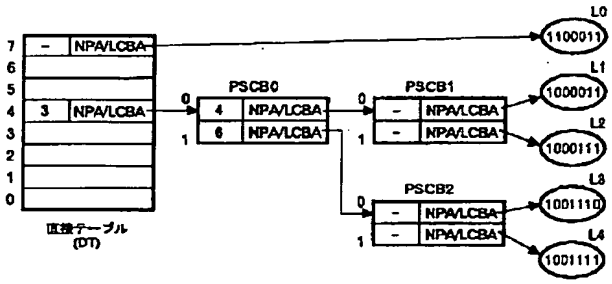
【図２】


オン・チップ・メモリ
オフ・チップ・メモリ
54
H0
H1
H2
H3
H4
S
D0
D1
D2
D3
D4
D5
D6
制御メモリ・アービタ
52
イングレス・エンキュー
完了ユニット
46
エグレス・エンキュー
割込み
EPCフリーズ
例外
デバッグ、割込み&シングル・ステップ制御
ハードウェア・レジスタおよびテーブル
内部制御バス
埋込み型パワーPCプロセッサ
PCIインターフェース
42
イングレス・データ記憶装置
プロトコル・プロセッサ
50
エグレス・データ記憶装置
44
48
命令メモリ
分類ハードウェア補助機構
バス・アービタ／バス・ウォッチ
内部バス
バス・ウォッチ
ディスパッチャ
50

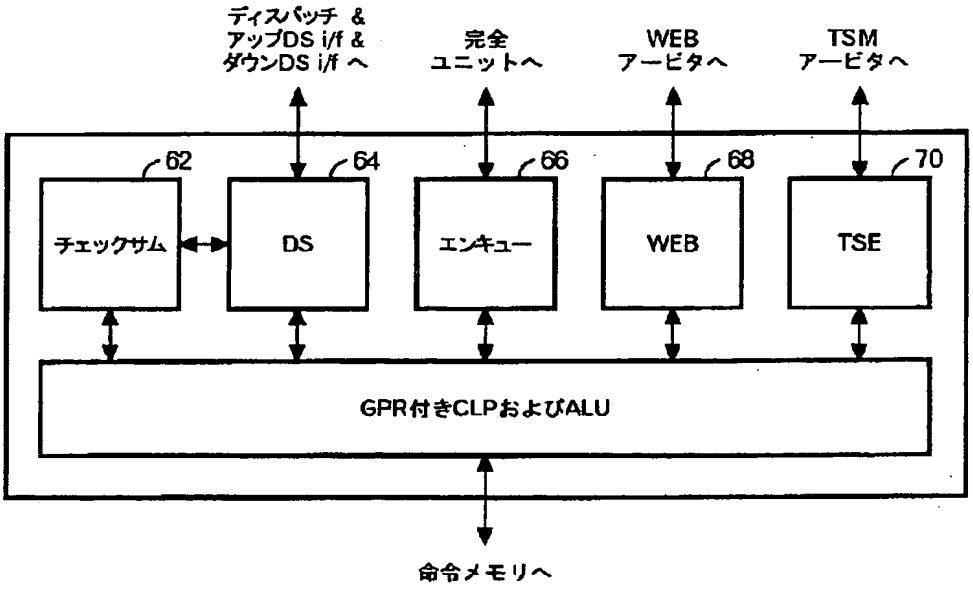
【図７】


108
リーフ
130
直接リーフのないデータ構造
直接リーフのあるデータ構造

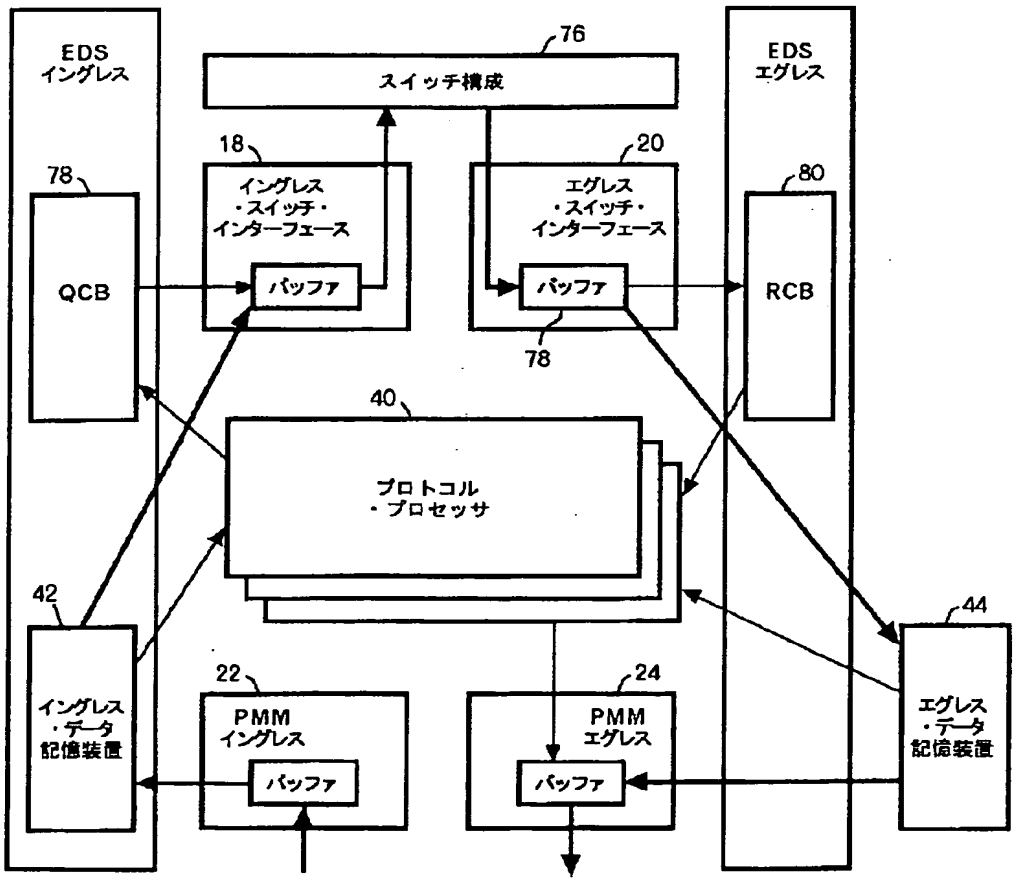
【図９】


NPA/LCBA
PSCB0
PSCB1
PSCB2
直接テーブル (DT)
L0
L1
L2
L3
L4
1100011
1000011
1000111
1001110
1001111

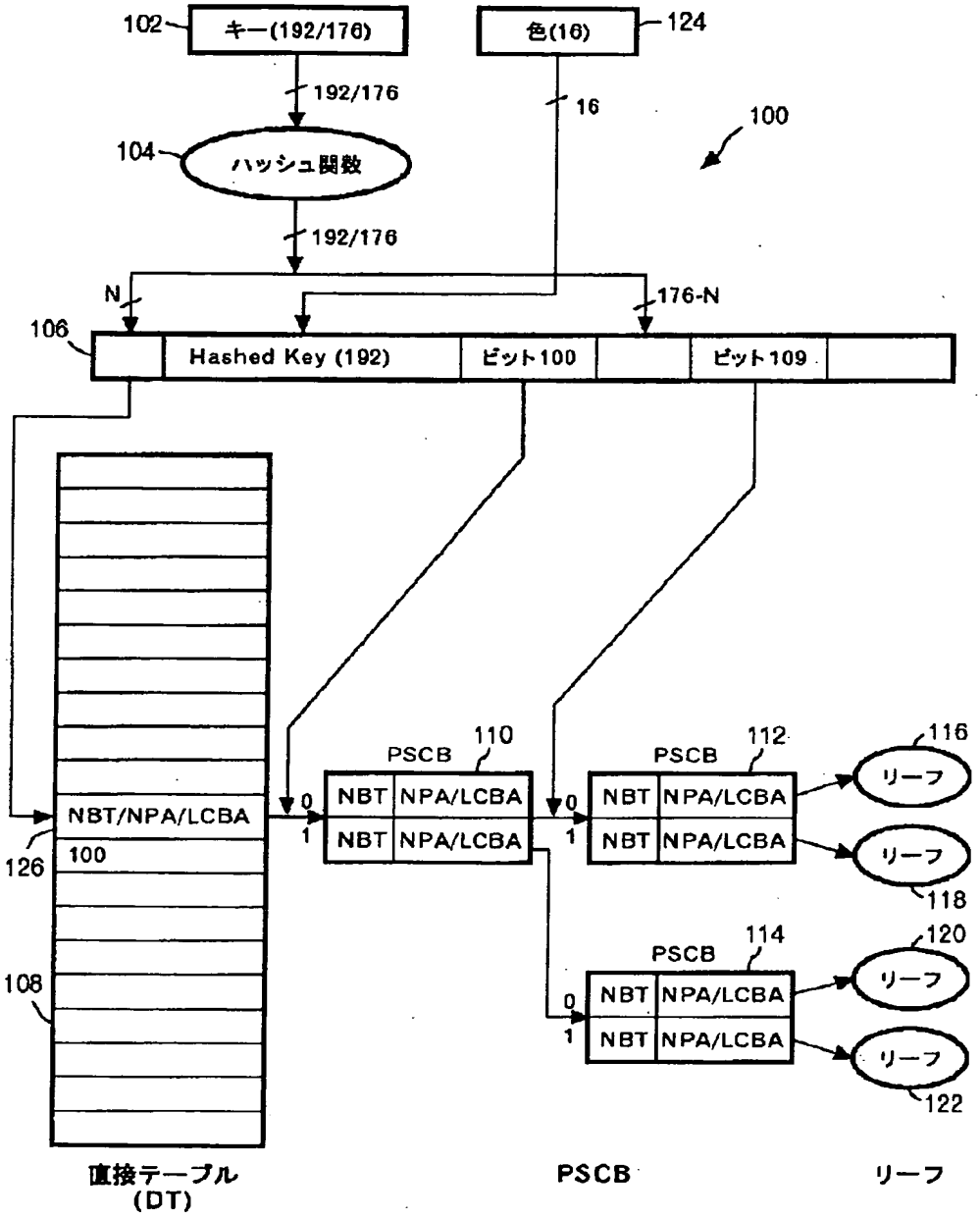
【図３】


ディスパッチ &
アップDS i/f &
ダウンDS i/f へ
完全
ユニットへ
WEB
アービタへ
TSM
アービタへ
62
64
66
68
70
チェックサム
DS
エンキュー
WEB
TSE
GPR付きCLPおよびALU
命令メモリへ

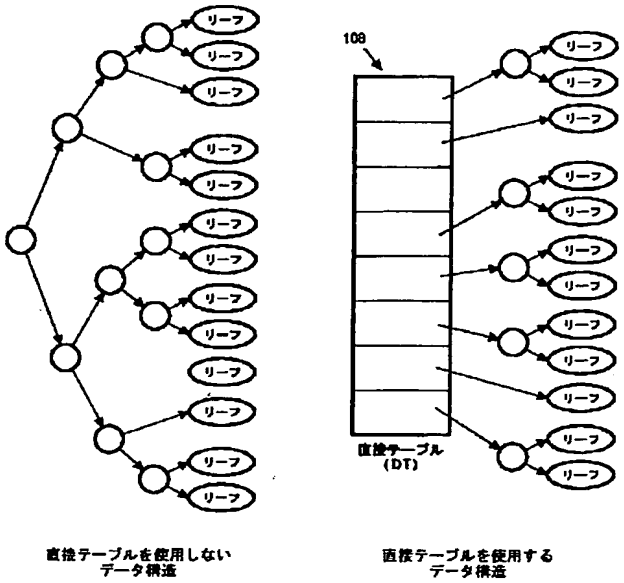
【図４】


EDS
イングレス
76
スイッチ構成
EDS
エグレス
18
イングレス
・スイッチ・
インターフェース
20
エグレス
・スイッチ・
インターフェース
78
QCB
バッファ
バッファ
80
RCB
78
40
プロトコル
・プロセッサ
42
44
イングレス
・データ
記憶装置
22
PMM
イングレス
バッファ
24
PMM
エグレス
バッファ
エグレス
・データ
記憶装置

【図５】


102
キー(192/176)
色(16)
124
192/176
16
100
104
ハッシュ関数
192/176
N
176-N
106
Hashed Key (192)
ビット100
ビット109
PSCB
110
PSCB
112
116
NBT/NPA/LCBA
0
1
NBT
NPA/LCBA
リーフ
126
100
118
PSCB
114
120
108
122
直接テーブル
(DT)
PSCB
リーフ

【図６】



【図１１】

LUDefTableの３つの定義

フィールド	サイズ	ビット
CacheEntry	1	0
Tree_Type	2	2..1
hash-type	4	6..3
color_en	1	7
P1P2_max_size	5	12..8
NPARope_en	1	13
NPASMT_en	1	14
CompIndex_en	1	15
PSCB_fq_index	6	21..16
PSCB_Height	1	22
Mask_Vector_En	1	23
CompIndex	8	31..24
DT_base_addr	26	57..32
DT_size	4	61..58
DT_interleaf	2	63..62
Leaf_fq_index	6	69..64
Leaf_Width	2	71..70
Leaf_Height	3	74..72
DirectLeafEn	1	75

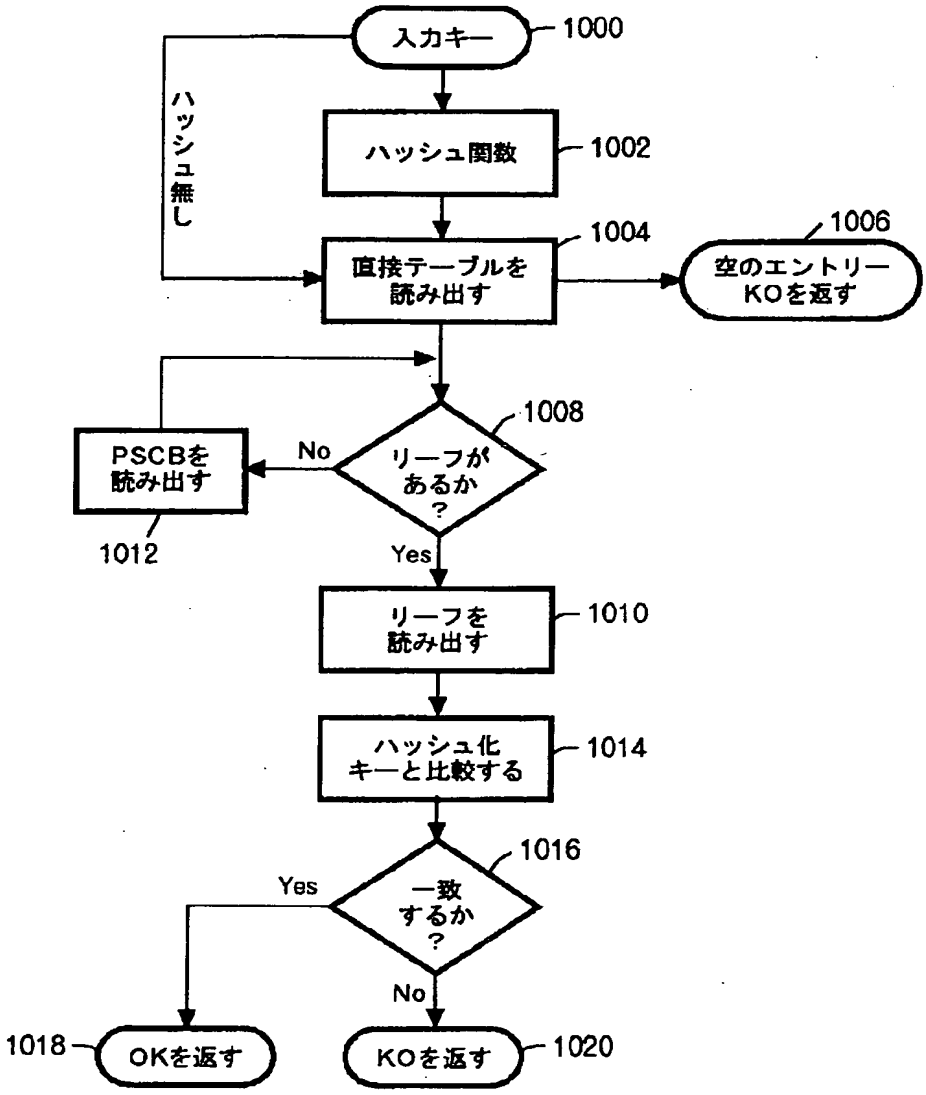
【図８】

フォーマット	状態	DTエントリで有効か？	DSCBで有効か？	フォーマット(2ビット)	NPA/LCBA(26ビット)	NBT(8ビット)
空のDTエントリ	リーフ無し	有効	無効	00	0	0
次のPSCBのポインタ	DTエントリはポインタを含む	有効	有効	00	NPA	NBT
リーフのポインタ	単一のリーフがDTエントリに関連付けられる；LCBAフィールドはポインタを含む	有効	有効	01	LCBA	0

【図１２】

フィールド	サイズ	PSCBが転置されるTSMのアドレス
NPA0	26	次のPSCBアドレス： PSCBの0部分のツリーの次のPSCBのポインタ
NBT0	8	PSCBの0部分の次の試験ビット
NCBA0	26	リーフ制御ブロックのアドレス： PSCBの0部分のリーフのポインタ
NPA1	26	次のPSCBアドレス： PSCBの1部分のツリーの次のPSCBのポインタ
NBT1	8	PSCBの1部分の次の試験ビット
LCBA1	26	リーフ制御ブロックのアドレス： PSCBの1部分のリーフのポインタ
Index	8	このPSCBのインデックス (物理的に前PSCBに格納される)
PatBit	1	PSCBレジスタ内のインデックス・フィールドの値に基づくHashedkey[インデックス]の値

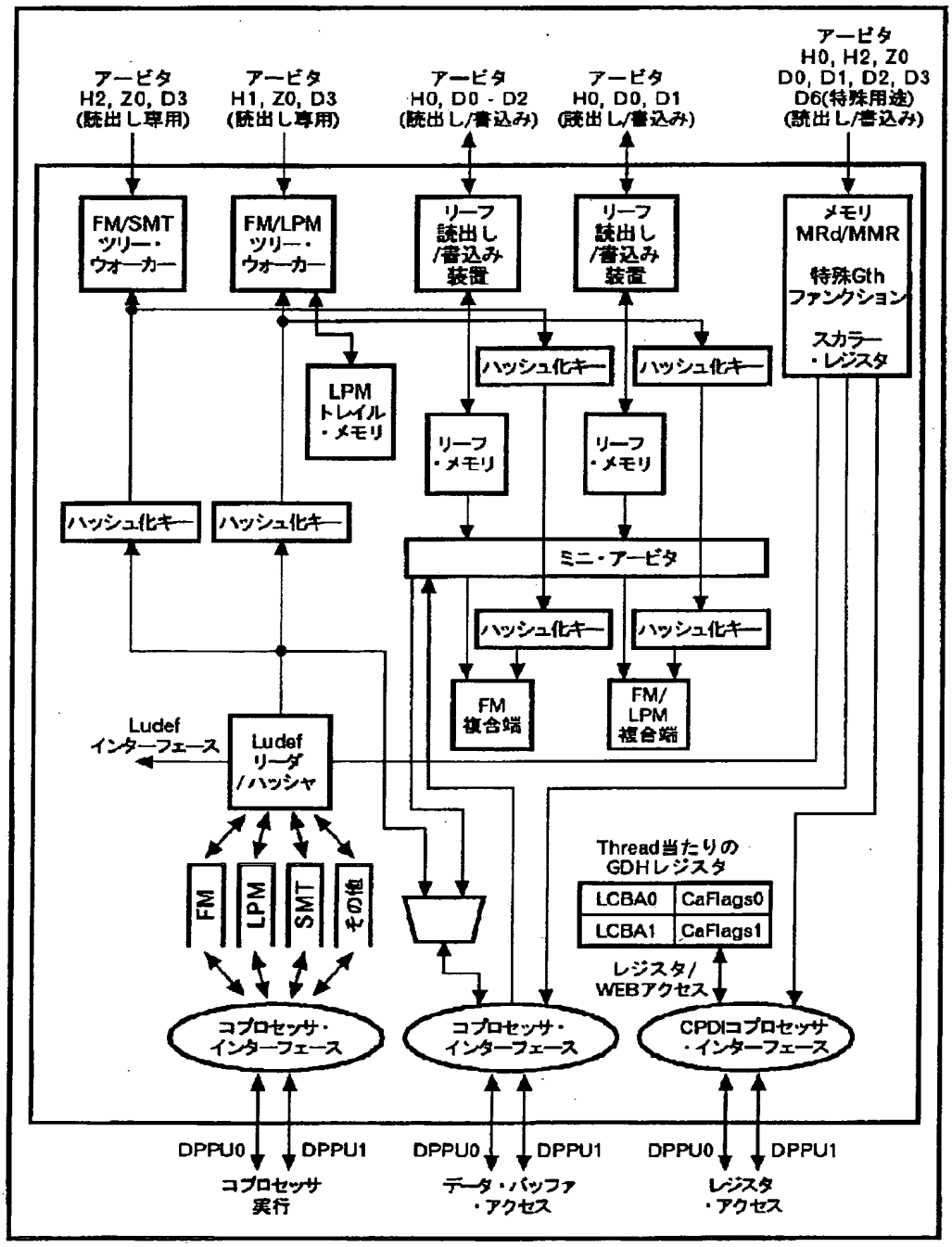
【図１０】



【図１３】

フィールド名	長さ	内容
NLARope	4バイト	リーフ連鎖ポインタ・エージング情報、および直接リーフ情報
Prefix_Len	1バイト	このフィールドはFMツリーの場合TSFによって使用されず、ピココードによって使用することができる
Pattern	2 - 18バイト	Hashedkeyと比較されるパターン
UserData	可変長	このフィールドの内容は完全なピココードの制御下にある；UserDataフィールドは１つまたはそれ以上のカウンタを含むことができる

【図１４】



フロントページの続き

(72)発明者　ブライアン・ミッチェル・バス
アメリカ合衆国27502　ノースカロライナ州アペックス　オールド・スタブリッジ・ドライブ4021

(72)発明者　ジャン・ルイ・カルヴィニャク
アメリカ合衆国27511　ノースカロライナ州ケアリ　スプリング・ホロー・レーン112

(72)発明者　マーコ・シー・ヘデス
アメリカ合衆国27612　ノースカロライナ州ローリー　グランド・マナー・コート4109　ナンバー308
(72)発明者　アントニオス・マラグコス
アメリカ合衆国27612　ノースカロライナ州ローリー　プリンセトン・ミル・パークウェイ3321　アパートメント201
(72)発明者　マイケル・スティーブン・スィーゲル
アメリカ合衆国27615　ノースカロライナ州ローリー　ラウリー・ドライブ10625
(72)発明者　ファブリス・ジャン・ヴェルプランケン
フランス06610　ラゴード　ルート・ド・カーニュ9152